COMP 1005J
Introduction to program construction 2

Henry McLoughlin

Module Details

- Double lecture every Monday 13:30 - 15:05
- Practical every Tuesday 15:25 - 17:00 starting week 2.
- During practicals I want you to work in groups, so let us form groups each with 9 or 10 students.
- This is very important, please get a notebook for use in practicals.

Module Details

- Assessment

- Final written exam 60%

- Practicals 40%

- We will have 4 practicals which will be graded and will contribute to the 40%, for the other practicals we will provide you with feedback so you can learn from them.

Online material

- The lecture slides will be available on the moodle server
- https://csmoodle.ucd.ie/moodle/course/view.php?id=774
- There is no password required.
- I may add other material too, but the material here isn’t the whole course, you must attend and take your own notes.

Contacting me

- You are welcome to contact me by email but please remember the following
- Send email to henry.mcloughlin@ucd.ie
- Send from your UCD email account not one of your other emails

A little bit about me

- 30+ years lecturing in UCD.
- Have taught in Ireland, China, Sri Lanka, Kenya.
- Have taught in China for 13 years (Fudan, Wuhan, BDIC)
- Extensive industrial experience (have founded 4 companies).
- Director of BSc.S/W Eng in BDIC.
- Major research interests “how to program and how to teach people to program”.

A Very Important fact

- Programming.....solving a problem and writing an algorithm is the difficult part of our work

- “Programming” ....coding our algorithm into a programming language like C or Java or Python is the easy part of our work

- Many people confuse there two parts of the work, I want you always to keep them separate.

Another important fact.

- Although programming (constructing an algorithm to solve a problem) is difficult, if you approach it calmly and slowly and think in a particular way, then it becomes easier.

- I don’t expect you to know how to think in this particular way yet, it is my job to teach you how to do so.

- But you must be prepared to learn and to forget some of your old ways of thinking. At times you may think I am going too slowly, but please relax and trust me, I know what I am doing :-)

An Apology

Some of you may have had previous experience programming, or you may just have a talent for it. If this is true then you may find that I am going too slowly.

This is quite deliberate, I want us to go slowly and observe our thinking when we solve problems so we can try to learn the best ways to approach problems.

Don’t worry if we are slow now, we will increase speed later.

Also, please follow the advice I give you in the module, and the rules I give you.

Review of what we should know: variables

- Variables are the names of places we use to store values.
- We can put values into variables
- We can use the values in variables
- We can change the values in variables

Review of what we should know: variables

- Variables should have a name
- Variables also have a type. This describes the sort of values that you can store in the variable. Examples might be integers, real numbers, characters, strings, boolean values etc.
- In Python you don’t have to state the type of the variable but in C you do have to.

Review of what we should know: variables

In Python.

```
x = 10
y = 20
print (x + y)
z = 2*x + y
```

In C

```
int x = 10;
int y = 20;
printf(“%d”, x + y);
z = 2*x + y
```

Review of what we should know: Selection

- In programming we often want to select different actions depending on some conditions.

- Most programming languages provide an “If statement” to allow you to make the choices.

- Suppose we had 2 integer variables x and y and we wanted to give integer variable z the values of the maximum of x and y.

Review of what we should know: Selection

In Python

```
if x <= y :
    z = y
elif y < x :
    z = x
```

In C

```
if (x <= y)
    {
      z = y;
    }
else if ( y < x )
    {
      z = x;
    }
```

Review of what we should know: Selection

In Python

```
if x <= y :
    z = y
else :
    z = x
```

In C

```
if (x <= y)
      {
        z = y;
      }
else
      {
        z = x;
      }
```

In these examples we don't write down the actual condition that decided whether we take the 2nd branch "y < x"
Some people prefer this as it saves time typing. I think it is usually better to actually write down why you entered a particular branch. It makes it easier to read the program.

Review of what we should know: Repetition

- In programming we often want to repeat some actions while some condition is true.

- Most programming languages provide a “While statement” to allow you to do this.

- Suppose you wanted to add up the integer values from 0 to 19 inclusive.

Review of what we should know: Repetition

In Python

```
i = 0
total = 0
while i != 20 :
    total = total + i
    i = i + 1
```

In C

```
int i = 0;
int total = 0;
while (i != 20)
      {
        total = total + i ;
        i = i + 1 ;
      }
```

Review of what we should know: Repetition

In Python

```
i = 0
total = 0
while i != 20 :
    total = total + i
    i = i + 1
```

The condition " i != 20 " is called the loop guard. When it is true the actions inside the loop are carried out. When it is false the loop finishes and the actions inside the loop are no longer carried out. At this point we know "i == 20"

Some people would have written the guard as " i < 20 ". When the guard is false and the loop finishes then we can say that "i >= 20 "

"i == 20 " is a much more definite fact and will prove to be much more useful to us in the future. So I would prefer you to write your guards like I do.

Review of what we should know: Repetition

- Many programming languages provide you with a number of different repetition statements. Examples include “Do loops”, “For loops”, “Repeat loops” etc....

- You should probably know enough about them to be able to read programs that use them, but I would prefer that you just use the “While loop” when we program together.

- Everything people can do with a For loop we can also do with a While loop, but not everything we can do with a While loop can be done easily with a For loop.

Review of what we should know: Repetition

In the C language there are a number of short cuts. Here is an example of one you quite often see.

```
int i = 0;
int total = 0;
while (i != 20)
      {
        total = total + i ;
        i++ ;
      }
```

Now i must decide whether to use
i = i +1
or
i++
or even something like
++i

So confusing!!
Programming is difficult enough
without us trying to make it more difficult.

So let is agree to just use i = i + 1

Review of what we should know: Functions

- In Mathematics we are familiar with the idea of a function.
- A Function takes a value (or a number of values) and produces a new value.
- We will give the function a name
- We will describe what type of values it will use
- We will describe what type of value it will produce for us.

Review of what we should know: Functions

- Many programming languages provide us with the ability to define and use “functions”

- These are not always like mathematical functions, sometimes they break some of the rules that apply to mathematical functions.

- However, they can be useful in helping us to decompose problems and often a function we write to solve one problem can be used in solving other problems.

Review of what we should know: Functions

Let us look at a simple function that takes 2 integers and
produces the largest of the two.

We will give the function a name and describe what type of values it takes and what type it produces.

max : Int X Int —> Int

In mathematics this is called the function signature. It tells us the name, the number and type of the inputs and the type of the output.

The actual definition of the function might be written like this

max (x, y) = x if x>= y
max (x, y) = y if x < y

Review of what we should know: Functions

In C

```
int max (int x, int y)
      {
  if ( x <= y )
       {
         return y;
       }
  else if ( x > y )
       {
         return x;
       }
}
```

In Python

```
def max (x, y) :
    if x <= y :
        return y
    elif x > y :
        return x
```

Review of what we should know: Functions

Suppose we wanted to get the largest of 3 integer values which were stored in the variables a, b and c.

How would we do so?

largest = max(a, max(b, c))

We don’t have to write a new function to take in 3 numbers, we can use the old one that took in 2 numbers.

Some people would have written this like this

largest = max(a,b)
largest = max(largest,c)

Which way is nicer

Leap years

A leap year is a year which either a year that is a multiple of 4 but not a multiple of 100

or

it is a multiple of 400.

Suppose you have a variable y which holds a year number, could you write a function to determine if y was a leap year or not?

What would that function return?

Prime numbers

A **prime number** is one where the only divisors of the number are 1 and itself.

Could you write a function that determined whether a number n was a prime number?

A **perfect number** is one where the sum of its divisors (other than the number itself) equals itself. 6 and 28 are examples of perfect numbers.

6 = 1 + 2 + 3
28 = 1 + 2 + 4 + 7 + 14

Could you write a function to determine whether a number was perfect?

Would these 2 functions have anything in common?