Introduction to C

Session 3

March 25

Dr Mark Doughty, School of Engineering and Physical Sciences,
University of Lincoln.

Code ‘blocks’

- A single line statement ends with a semicolon:

```
int x = y + z;
```

- Multiple statements:

```
int x = y + z;
int q = aFunc(x);
```

- We can use curly braces to combine these into a ‘block’

```
{
     int x = y + z;
     int q = aFunc(x);
}
```


Dr Mark Doughty, School of Engineering and Physical Sciences, University of Lincoln.

Code ‘blocks’

- Variables can be declared inside or outside code blocks:

```
int q = 0;
{
    int x = y + z;
    q = aFunc(x);
}
```


Dr Mark Doughty, School of Engineering and Physical Sciences, University of Lincoln.

Code ‘blocks’

- We can have blocks inside other blocks – nested blocks:

```
int q = 0;
{
     int x = y + z;
     q = aFunc(x);
     {
          float x1 = y*z;
     }
}
```


Dr Mark Doughty, School of Engineering and Physical Sciences, University of Lincoln.

Scope

- ‘Scope’ indicates the availability of a variable – i.e. where it is valid
- If a variable is created inside a block – it’s scope or availability is only within that block (or blocks nested within it)
  - Its scope is ‘local’ to that block
- For example:
  - ‘x1’ is only available within the inner block*
  - ‘q’ is available to all blocks and wider*

*From their point of declaration to end of their containing block

```
int q = 0;
{
        int x = y + z;
        q = aFunc(x);
        {
                float x1 = y*z;
        }
}
```


Dr Mark Doughty, School of Engineering and Physical Sciences, University of Lincoln.

Program Control

- Control flow conditions in your program mean that it can make decisions based on some control values
- Conditions are controlled by Booleans (true/false)
  - Use <stdbool.h>
- Booleans are set by conditional statements
  - `if` statements
  - `switch` statements

Dr Mark Doughty, School of Engineering and Physical Sciences, University of Lincoln.

if statement

```
int x = 5;
if(x < 10)
    printf("%d is less than 10", x);
```

The indentation is not needed, but is really useful to help readability.

It is added automatically by VS Code.

- First, do the test and evaluate the condition
  - Is `x < 10` ?
    - If it is true – do the following line (the printf statement)
    - If it is false – miss the following line and skip to the next.

Dr Mark Doughty, School of Engineering and Physical Sciences, University of Lincoln.

if statement – with a code block

```
int x = 5;
    if(x < 10){
        printf("%d is less than 10", x);
        x++;
    }
```

One school of thought is to always use braces – even for one line.

Good for program readability and understandability.

- First, do the test and evaluate the condition
  - Is `x < 10` ?
    - If it is true – do the following code block (code between the braces)
    - If it is false – miss the following code block and skip to the next line.

Dr Mark Doughty, School of Engineering and Physical Sciences, University of Lincoln.

if - else

```
int x = 5;
if(x < 10){
    printf("%d is less than 10", x);
    x++;
}else{
    printf("%d is greater than 10", x);
}
```

- First, do the test and evaluate the condition
  - Is `x < 10` ?
    - If it is true – do the following code block, then skip to the first statement after the `if`
    - If it is false – do the code block after the '`else`', then skip to the first statement after the `if`

Dr Mark Doughty, School of Engineering and Physical Sciences, University of Lincoln.

else if

```
int x = 10;
if(x < 10){
    printf("%d is less than 10", x);
    x++;
}else if (x > 10){
    printf("%d is greater than 10", x);
}else{
    printf("%d is equal to 10", x);
}
```

This is like a 'nested `if` statement' – `if` statements within another `if`.

- Do the first test and evaluate the condition
  - Is `x < 10` ?
    - If it is true – do the following code block, then skip to the first statement after the `if`
    - If it is false – test the `else if` condition, if it is true, do the following block, if false, do the last `else` block.

Dr Mark Doughty, School of Engineering and Physical Sciences, University of Lincoln.

Nested if

```
int x = 5;
if(x < 10)
    if(x == 10)
        printf("%d is equal to 10", x);

else
    printf("well I never");
```

Be careful to use braces to make it clear what you mean.

To which ‘if’ statement does the ‘else’ belong here?

```
int x = 10;
if(x <= 10){
    if(x == 10)
        printf("%d is equal to 10", x);
}
else{
    printf("well I never");
}
```

Make sure the conditions to be tested allows the program control to go along the path you expect…

If the first test wasn’t ‘<=‘, then the inner if wouldn’t be tested.

Dr Mark Doughty, School of Engineering and Physical Sciences, University of Lincoln.

switch

- Has to be an integer or character as input to test.
- Test cases are the ‘case’ statements.
- If a match is found, statements are executed until the `break;`
- After the `break;` control flow moves to first statement after the `switch`.
- Without the ‘`break;`’ control ‘falls through’ to the next statement.
- ‘`default`’ is needed to provide a case for when all others are not met.

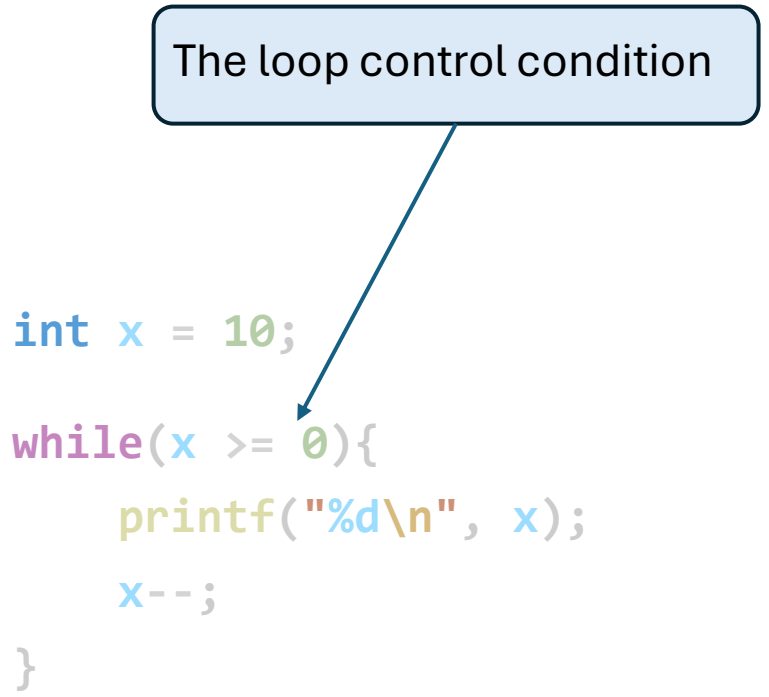
```
int x = 10;
switch(x){
    case 10:
        printf("%d is equal to 10",x);
        break;
    case 0:
        printf("%d is equal to 0",x);
        break;
    default:
        printf("%d is neither 0 or 10",x);
        break;
}
```


Dr Mark Doughty, School of Engineering and Physical Sciences, University of Lincoln.

Loops – the `while` loop

- This loop runs (i.e. it executes the code in the loop body) as long as the condition is true.

- If the loop control never becomes false, then this is an infinite loop.



In this loop, x is reduced by 1 each time the loop runs until the condition is false.

The condition is tested before each loop runs – a '**prefix**' test. The loop may never run if the condition starts as 'false'.

Dr Mark Doughty, School of Engineering and Physical Sciences, University of Lincoln.

Loops – the for loop

- A for loop uses three expressions separated by ';'

```
          1          2     3
for(int x = 1; x <= z; x++)
```

- 1: Declare and initialise loop control variable.
- 2: Loop control conditional test.
- 3: Increment/decrement of the loop control variable.

```
int y = 1;
int z = 10;

for(int x = 1; x <= z; x++){
    y *= x;
}

printf("%d factorial is %d", z, y);
```


Dr Mark Doughty, School of Engineering and Physical Sciences, University of Lincoln.

Loops – the `for` loop

- The loop control variable scope is **only** within the loop block.
- If the conditional test (2) never gets false – then this is an infinite loop.
- All three control expressions can be blank (still need the ';' though)
  - This is the same as saying '`while(true){…}`'
  - `for( ; ; ){ … }`
  - This will then continue to run (infinite loop), unless a '`break;`' ends the loop.

Dr Mark Doughty, School of Engineering and Physical Sciences, University of Lincoln.

Loops – the `do - while` loop

- This does the same as the `for` loop previously.
- This is a '**postfix**' loop
  - The loop control condition is tested after the loop has run once
  - The loop is guaranteed to run at least once.
  - Ends with a ';'

```
int y = 1;
int z = 10;
int x = 1;

do{
    y *= x;
    x++;

}while(x <= z);

printf("%d factorial is %d", z, y);
```


Dr Mark Doughty, School of Engineering and Physical Sciences, University of Lincoln.

break

- Sometimes we want to end a loop early.
- `break;` forces the loop the end at that point
  - Control moves to the first statement after the loop.

```
int y = 1;
int z = 10;
int x = 1;

do{
    y *= x;
    x++;
    if(x == z)
        break;

}while(true);

printf("%d factorial is %d", z, y);
```


Dr Mark Doughty, School of Engineering and Physical Sciences, University of Lincoln.

continue

- Use to skip one iteration at that point.
- In the example, x is iterated, then tested to see if it is even
  - If it is, the loop returns to the start of the loop for the next iteration – missing out the rest of the loop body.

```
int y = 0;
int z = 10;
int x = 0;

do{
    x++;
    if(x % 2 == 0)
        continue;
    y += x;
}while(x < z);

printf("Sum of odd numbers is %d", y);
```


Dr Mark Doughty, School of Engineering and Physical Sciences, University of Lincoln.

In conclusion …

- In this session, we have covered:
  - Statement blocks
  - Scope
  - Program control flow
    - `if, if .. else, elseif`
  - Loops
    - `while, do .. while., for`
  - Loop control
    - `break`
    - `continue`

Dr Mark Doughty, School of Engineering and Physical Sciences, University of Lincoln.