




# “Get Techy”

---

## Trinity Hall JCR Programming Club Session 3

*Sinéad McAleer - Tech Officer*  
*[mcaleesi@tcd.ie](mailto:mcaleesi@tcd.ie)*



# Last Week

---

- Finished off Scratch
- Intro to C
- Variables
- Data Types
- Wrote IF/ELSE & printf statements

## Challenge:

- *Umbrella Calculator:*
- Write a program which determines if the user should bring an umbrella when they go out (if it looks like it might rain or is raining) and if so whether they should put it up (if it is raining).
- What you will need:
  - A Boolean for whether or not it is raining and whether or not it looks like it will rain (in C booleans are simply integers that are 1 or 0)
  - You will need an IF/ELSE statement
  - You will need two printf statements
  - You can decide whether or not it is raining! Change it up and see if it works for all possible outcomes.



C



Week 2



# Solution to last week:

---

```
#include <stdio.h>
```

This is the library for standard input and output

```
int main()  
{
```

int main() means it is the MAIN program we are running.  
In our case, it contains our entire umbrella program

Our curly bracket acts as a block to contain our main

```
int isRaining = 0;  
int looksLikeRain = 1;
```

We are defining our two boolean variables.

# Last Week:

---

Last week we defined variables like

```
age = 15;
```

A boolean (a value which is true or false, such as awake or isRaining) is simply an integer with the value 1 or 0.

0 is false. 1 is true. The default value is 0/FALSE.

# Solution to Last Week:

---

```
if(isRaining == 1 || looksLikeRain == 1)
```

```
{  
    printf("Bring an umbrella!\n");  
}  
else  
{  
    printf("There is no need to bring an  
umbrella...probably.\n");  
}  
}
```

This is our IF statement. The brackets contains our conditions.

These conditions say "IF isRaining is true OR IF looksLikeRain is true then..."

"if(...)" means if

"||" means OR

"==" means equals

# Solution to Last Week:

---


```
if(isRaining == 1 || looksLikeRain == 1)
{
    printf("Bring an umbrella!\n");
}
else
{
    printf("There is no need to bring an
umbrella...probably.\n");
}
```

These sets of curly brackets contain the printf statement we want to print to the console.

# Solution to Last Week:

---

```
if(isRaining == 1 || looksLikeRain == 1)
{
    printf("Bring an umbrella!\n");
}
else
{
    printf("There is no need to bring an
umbrella...probably.\n");
}
}
```



Q: What is this curly bracket in relation to?

A: The curly bracket on the first slide under the main() statement.



# Remember!

---

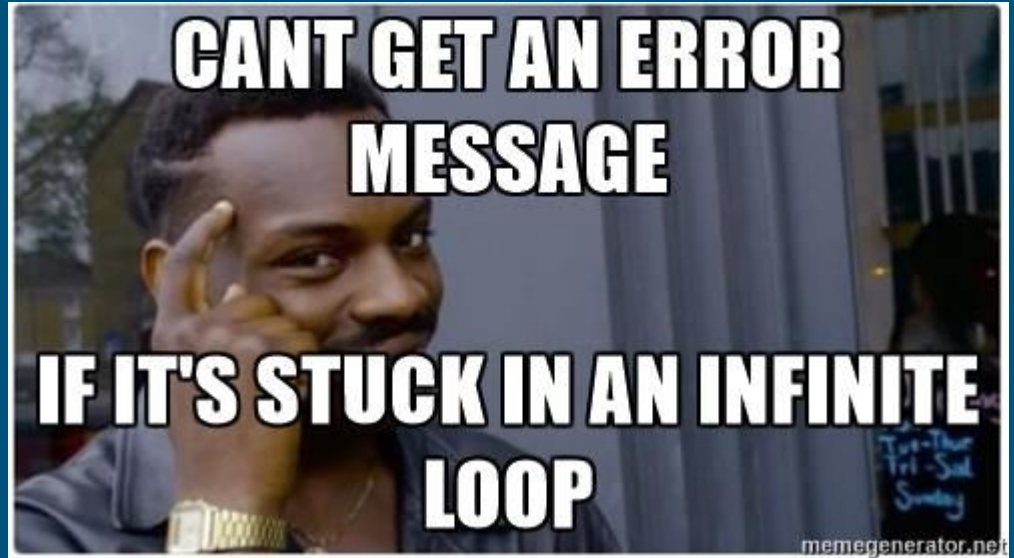
After the “if(...)” we do not use a semi-colon, as we are not done our “sentence” yet.

After printf statements and declaring variables (“int age = 15;”) we use a semi-colon as we want to indicate to the computer to take a new line.

# This week!

---

- Loops



# Feeling Loopy

- When we worked on scratch we were introduced to loops.
- Three loops you may remember are:

We used forever to make our sprite walk until we closed the program



We used repeat 10 for our dancing



You could use repeat until for something like - repeat until score is higher than 10

# We are now going to look at these in C

---

```
int i;  
for(i = 0; i < 10; i++)  
{  
    printf("Keep dancing\n");  
}
```



This is a for loop. There's a lot going on here so let's take it piece by piece.

# For

---

“for” is simply the name of the loop.

(i = 0... This means that we declare an integer as 0. We name it i as this looks good, but we could just as easily call it count. This is a temporary variable, we are just using it to keep track of how far into our looping we are.

...i < 10... In the same way that we repeated 10 we just want to loop through this loop 10 times. Therefore, while our counter i is less than 10 we continue looping.

```
int i;
for(i = 0; i < 10; i++)

{

    printf("Keep
dancing\n");

}
```

# For

---

...i++)... This is a quick and easy way of saying  
`i = i + 1;`

{... Remember, { and } are braces which contain sections of code. They contain the code we want to run in our looping.

```
int i;  
for(i = 0; i < 10; i++)
```

```
{  
  
    printf("Keep  
    dancing\n");  
  
}
```

# For

---

What this block of code means:

Loop through the printf statement 10 times.

This will result in the statement being printed 10 times

You may wonder *why* we would want to do this...

So this brings us to our first Challenge.

```
int i;  
for(i = 0; i < 10; i++)  
  
{  
  
    printf("Keep  
dancing\n");  
  
}
```

# Challenge #1

---

- On CodeBoard.io
- Print out the square number of every number up to 16.

HINTS:

Remember this is how we declared our for loop:

```
int i;  
for(i = 0; i < 10; i++){  
  
}
```

This time you will have to keep  
stdlib.h

HINT:

You can use the int i in calculations inside the for loop. For example, you may wish to put

```
int SquareNumber = do something with i
```

Then you will want to print the squareNumber.  
Try googling how to print an int using printf

Remember: You must compile before you run to  
update any changes

<https://codeboard.io/projects/63248>



# Solution

---

```
int main()
{
    int i;
    for(i = 0; i < 16; i++)
    {
        int squareNumber = i*i;
        printf("%d\n", squareNumber);
    }
}
```

Can you recall what `int main()` is?

These lines are your for loop.

Here is a bit of maths! The `squareNumber` is just `i` multiplied by itself. You may have found other ways to do this....

The `printf` is a bit different as this time we are printing a **variable**, not just words.

# While Loops

---

While loops are even easier.

They look like this

```
while(age < legalDrinkingAge)

{ ...
```

The block of code within the brackets will simply be repeated until the condition is met. When the condition is met we **break** out of the while loop.

What if the condition is never met?

Then we are stuck in an infinite loop and we should really do something to fix that.

# Example of Code:

---

```
int main()
{
    int age = 14;
    int legalDrinkingAge = 18;
    while (age < legalDrinkingAge)
    {
        printf("Not legal to drink \n");
        age++;
    }

    printf("Now you are %d you are legal to
drink!\n", age);
}
```

<https://codeboard.io/projects/64814>

This statement will be printed every time we loop through the while loop.

This statement will only be printed after we have broken out of the while loop and are moving on with the program.

# Challenge #2

---

Complete the same program as Challenge #1 but this time with a while loop.

- Print out the square number of every number up to 16.
- For while loop you will need to introduce some sort of count.

While statements look like:

```
while ( x > y){  
  
    **do something**  
  
}
```

<https://codeboard.io/projects/64985>

# Solution

---

```
#include <stdio.h>
#include <stdlib.h>
```

```
int main()
{
    int count = 0;
    while(count < 16)
    {
        int squareNumber = count * count;
        printf("%d\n", squareNumber);
        count++;
    }
}
```

# Recap

We now know:

- How to declare variables
- How to do simple maths
- How to do IF/ELSE
- How to do for/while loops



# Challenge

---

Pick a random number (eg. 100). Write a program that prints all the **even** numbers up until this number. At the end print the number of even numbers found.

You will need:

- A for/while loop
- A count of values printed
- The modulo (%) figure. Use google to figure out what this means in C!

HINT: Always define variables at the top of your program. If you define them within a for/while loop you will not be able to use them outside of this loop.

For extra credit (and a headstart on the Week 6 Challenge) send this code to me at some point before next Wednesday!