“Get Techy”

Trinity Hall JCR Programming Club
Session 4

Sinéad McAleer - Tech Officer
mcaleesi@tcd.ie

Last Week

- Revised IF/Else
- Revised declaring variables
- Did some simple maths
- For loops
- While loops

Challenge:

- *Printing Even Numbers:*
- 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!

C

Week 3

Feeling Loopy - Revision

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

We used repeat 10 for our dancing

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

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

While Loops

They look like this

```
while(age < legalDrinkingAge)

{ …
```

This block of code within the brackets will simply be repeated while the condition is **true**. When the condition is false we get 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.

For Loop

"for" is simply the name of the loop.

(i = 0;... This means that we declare an integer as 0. We name it *i* inspired from Maths (Summation Σ or Matrix A(i, j)...) but we could just as easily call it count. If declared inline (for (**int** *i* = 0; it can only be used inside loop.

...i < 10; The second statement checks if the loop should run each time. In this case, if value of i is less than 10 it will run.

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

{
...
this block repeats for
each value of i less than
10
...
}
```

For Loop

i++)... The third statement runs each time it reaches end. i++ increments value by one. It's a quick way of saying

```
i = i + 1;
```

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

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

{
...
this block repeats for each
value of i less than 10
...
}
```

The Empty Statement

;

Forever Loops

We lied, those three statements in for loop are optional! You can replace it with empty statements.

```
for ( ; ; )

{    …

runs until there's love in this world i.e. forever

}
```

```
while (1)          i.e. while (true)

{    …

or until you press Ctrl + C

…

}
```

To exit the loop you have use **break;** statement inside the block.
These are useful when you don't know the condition to end or have multiple exit points.

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.

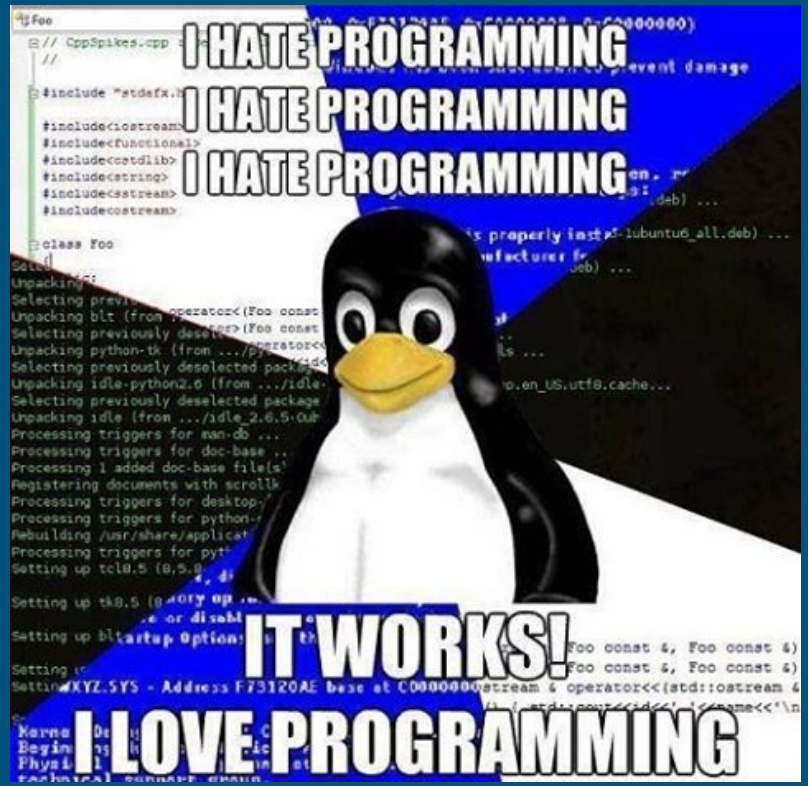
<- This statement will only be printed after we got out of the while loop and are moving on with the program.

Last Weeks Challenge

Printing Even Numbers:

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!

https://repl.it/@dsmudhar/Solution-to-Week-3



Solution to Last Week

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h> //time header

int main(void)
{
   srand(time(NULL));

   int randomNumber = rand() % 99 + 1;
   printf("Random Number: %d\n",
randomNumber);
```

These first three lines are our libraries.

Our main() { .. } is our main chunk of code which is ran by system

<- srand() initialized our rand() function by using current time.

<- rand() gives us random number

What does the “//” mean? 

Solution to Last Week

```
    for(int i = 1; i < randomNumber; i++)
    {
        if (i % 2 == 0) {
            printf("%d ", i);
        }
    }
    printf("\n");
}
```

This is our for loop.

The modulo figure calculates the remainder of a division.

Remember, curly brackets contain chunks of code relating to certain IF statements/for loops

This Week!

- User input
- More calculations
- Formatting your code

More Control Flow

```
if (var == VAL_1) {
...
} else if (var == VAL_2) {
...
} else if (var == VAL_3) {
...
} else {
...
}
```

```
switch (var) {
        case VAL_1:
        runs these statements and consecutive
ones
case VAL_2:
        this one runs;
case VAL_3:
        this one also;
        Break; //until break statement is used.
Default:
        //if there’s no case defaut one works
}
```

User Input

- Very often we want to take input from our user
- We may want to ask a user their name, their age, whether they want to play a one-player or two-player game, etc
- The scanf() function is the input method equivalent to the printf() output function - simple yet powerful.

Using scanf()

```
#include <stdio.h>

int main(void)
{
    int input;
    printf("Please input an integer value: ");
    scanf("%d", &input);
    printf("You entered: %d\n", input);
}
```

When we take input from the user, we need somewhere to store it. Therefore, we define a variable.

printf(...); - We tell the user what we want them to enter. We must be as clear as possible. We don't want them to enter a letter when we are expecting a number.

scanf("%d", &input); - this is a bit different. Let's look at it carefully.

Using scanf()

```
scanf("%d", &input);
```

As we saw last week %d indicates it is an int that we are dealing with.

But why do we use &input as opposed to just input when we are using printf?

The scanf() function requires the **memory address** of the variable to which you want to save the input value.

The & symbol is operator that gives “address of”.

https://repl.it/@jcrtech/Using-Scanf

Memory Addresses

- All the variables are stored in memory. By name (like age) we get its value but not where it's stored.
- You can imagine the memory like list of 1's and 0's with index. Each 32 of these make a integer (in binary).
- The pointer a is pointing towards the value stored in the address e.g. 876.
- When we assign a value to a variable, we are actually setting values to 1 or 0 etc. starting from address 876.

Pointer	Memory	Address
a →	17	876
	0	875
	0	874
	0	873

Memory Addresses

- So when we type

```
scanf("%d", &input);
```

- We are actually indicating that we want to store something at the address of input

Pointer	Memory	Address
input →	0	876
	0	875
	0	874
	0	873

Challenge #1

This week we are going to use a different IDE. Previously we used CodeBoard.io but we are now going to use repl.it

repl.it - Online REPL, Compiler & IDE

- It's good to practice with more than one IDE to transfer your skills
- If you want to save your work you have to create an account!

Challenge #1

Calculate a user's BMI based on their height (ms) and weight (kgs).

BMI = weight/(height^2)

Hints:

- You are going to have to use scanf("%d", &variable); twice
- To use decimal places we use "float"
- https://repl.it/@jcrtech/BMI

Solution

```
int main()
{
  int height;
  int weight;

  printf("Enter height: ");
  scanf("%d", &height);
  printf("Enter weight: ");
  scanf("%d", &weight);

  float bmi = height*height;
  bmi = weight/bmi;
  printf("BMI is %.6f\n", bmi);
}
```

Formatting your Code

- There are certain “norms” when it comes to coding
- You may have noticed some before in code done in these workshops
- For example,
- We take a new line for curly braces --->

```
if(age < legalDrinkingAge)
{
        printf(“You are not legal to drink.”);
}
else
{
        printf(“You are legal to drink.”);
}
```

Naming Variables

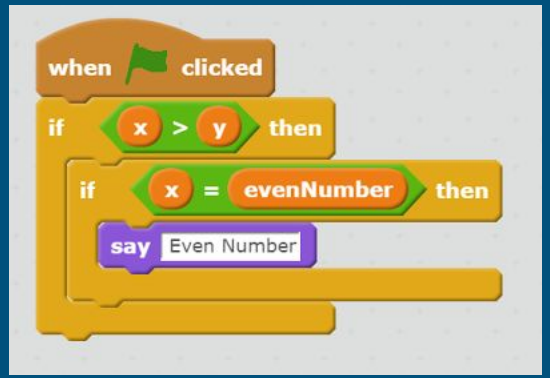
- When naming our variables they are coding conventions we normally abide by
- For example,
- One word variables are all lowercase --->
- Two word variables are camelCase or contains underscore e.g. age_2.
- Booleans usually starts with is, has e.g. is_age_legal.

```
int age = 15;
Int legalDrinkingAge = 18;
```

We do not take spaces in variables. We try to make them as clear as possible to increase understanding of our code so others can understand our code

Indenting Our Code

- If you remember in Scratch our yellow blocks neatly tucked inside each other - aka they were indented
- To do this we use the shift key for blocks of code contained within curly brackets




```
while(x > y)
{
        if(x%2 == 0)
        {
                printf("Even number");
                x++;
        }
}
```

Challenge #2 - Hi-Lo Game

- Randomly generate a playing card (playing cards are ints between 1 - 13) (for simplicity, Ace will be equal to 1)
- Display the card value
- Prompt the user to guess whether the next card will be 1) higher, 2) lower or the 3)same
- Tell the user if they were correct or not

To simplify the input, you can expect the user will enter the int 1 for higher, 2 for lower and 3 for the same.

For extra credit, allow the user to continue guessing until the enter quit. You will need a while loop for this.

For extra, extra credit allow the user to enter the int 1 or the string “higher” to indicate their guess. You will need to google how to use scanf(...); for string values!