PROGRAMMING

CT103

Week 3a

Lecture Content

- Last lecture (Week 2b):
  - Basic maths operators.
  - Modulus.
  - Else if statements.
  - Nested if statements.
- Today's morning lecture (Week 3a):
  - Boolean Logic.
  - Switch Statements.
  - Characters.

BOOLEAN ALGEBRA

Boolean Algebra

- We introduced Boolean logic last week.
- We saw AND which is && in C.
- We also saw NOT which is ! in C.

```
int temp = 35; // deg C
int rain = 0; // 0 = no rain, 1 = rain
if (temp > 18 && !rain) {
    printf("bring suncream \n");
}
else {
    printf("don't bring suncream \n");
}
```

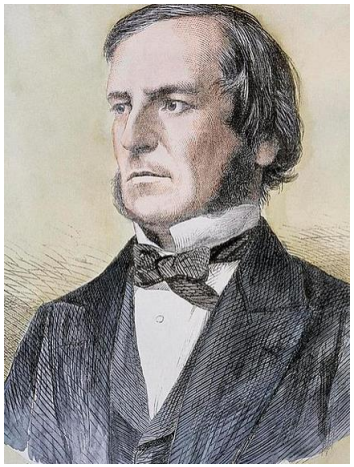
Boolean Algebra

- What is Boolean Algebra?
- Definition: Boolean Algebra is a form of algebra in which all variables are either True or False.
- Boolean operators can then applied to these variables.

George Boole

- Boolean algebra is named after George Boole who first introduced it.

- George Boole was a Professor in UCC, Cork Ireland.



George Boole
Image from: Wikipedia

Boolean Operators

- The primary Boolean operators are:
  - AND (In C: **&&**)
  - OR (In C: **||**)
  - NOT (In C: **!**)
  - XOR (In C: **!=**)

Truth Tables

- The following truth tables show how each of these operators work.
- In C: 1 = True, 0 = False

NOT

x	x'
0	1
1	0

AND

x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	$x+y$
0	0	0
0	1	1
1	0	1
1	1	1

XOR

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

Source: https://introcs.cs.princeton.edu/java/71boolean/

Boolean Operators in C

- AND
- What will the following code output?

```
int a = 1;
int b = 1;
if (a&&b) {
    printf("True \n");
}
else {
    printf("False \n");
}
```

Boolean Operators in C

- OR
- What will the following code output?

```
int a = 0;
int b = 0;
if (a||b) {
    printf("True \n");
}
else {
    printf("False \n");
}
```

Boolean Operators in C

- XOR
- What will the following code output?

```
int a = 1;
int b = 1;
if (a!=b) {
    printf("True \n");
}
else {
    printf("False \n");
}
```

Boolean Operators in C

- NOT
- What will the following code output?

```
int a = 0;
int b = 1;
if (!a) {
    printf("True \n");
}
else {
    printf("False \n");
}
```

SWITCH STATEMENTS

Switch statement

- Switch statements test the value of a variable and compares it with multiple cases.
- If case match is not found, default statement is executed.
- Benefits of switch statements:
  - Switch can be tidier.
  - Can be executed faster.

Switch Template

```
switch (expression)
{
    case value1:
     // do something
        break;
    case value2:
        // do something else
        break;
        ...
    default:
        break;
 }
```

Note : not ;

- Expression is evaluated.
  - **Expression** must return an int.
  - Expression can be an int.
- Value of expression compared to each case.
- *Break* important to avoid running on and executing the next case (if you leave it out, it will!)

Switch Example in C

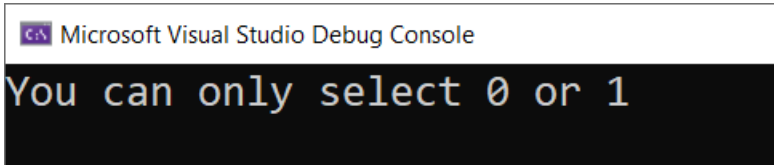
- Switch statement that checks if a number is 0 or 1.

```c
// switch statement
int num = 11;
switch (num) {
case 0:
    printf("You have selected 0\n");
    break;
case 1:
    printf("You have selected 1\n");
    break;
default:
    printf("You can only select 0 or 1\n");
    break;
}
```

Sample Output

- If we run the following statement with num =11, we get the default response.

```
// switch statement
int num = 11;
switch (num) {
case 0:
    printf("You have selected 0\n");
    break;
case 1:
    printf("You have selected 1\n");
    break;
default:
    printf("You can only select 0 or 1\n");
    break;
}
```



Equivalent Program using IF Else

- Below we can compare both programs side by side using If Else and using Switch.

```c
// equivalent program using if else
int num = 11;
if (num==0) {
    printf("You have selected 0\n");
}
else if (num==1) {
    printf("You have selected 1\n");
}
else {
    printf("You can only select 0 or 1\n");
}
```

```c
// switch statement
int num = 11;
switch (num) {
case 0:
    printf("You have selected 0\n");
    break;
case 1:
    printf("You have selected 1\n");
    break;
default:
    printf("You can only select 0 or 1\n");
    break;
}
```

CHARACTERS

Characters in C

- What are they really?
- How are they stored?
- How do we read them in?
- Hanging newline characters in the input
  - And how to get rid of them

How are variable values stored

- 1’s and 0’s – everything is stored in binary format.
- That includes characters also. Each character has a different binary value.
- char c = ’a’;
- Note: Singe quotations for characters.
- Other languages, e.g. python, are less strict with quotations.

What are the values behind the characters?

- This is where having a standard character table comes in.
- Enough people in industry got together and decided what the value of each character should be.
- So for example:
  - ‘a’ is stored as the number 97 (binary 1100001)
  - ‘A’ is stored as the number 65 (binary 1000001)
  - ‘?’ is stored as the number 63 (binary 111111)
  - ‘#’ is stored as the number 35 (binary 100011)
  - … and so on

  - The full set is called a character set, such as the original ASCII (American Standard Code for Information Interchange) table
  - Since superseded by UTF, but UTF includes the basic ASCII English character set

How to see the value of a character

- The following will show you the value of a character:

```
char myChar = '!';
printf("character \'%c\' represented by value %d \n",myChar, myChar);
```

```
character '!' represented by value 33
```

EXAMPLE C PROBLEM

Quality Control Program

- You are writing a computer program for a manufacturer to check if the quality of a product. Write a C program:
  1. Create two variables to store the width and height of a product in meters. Test using measurements: w = 0.21m, h = 0.15m.
  2. Convert the width and height to millimetres.
  3. Check if the product width is outside of the acceptable region. Min width = 200mm. Max width = 230mm.
  4. Do step 3. twice, first using OR, then using AND.
  5. Categorize the height as short, medium, or tall based on the table below:

Category	Min Height (mm)	Max Height (mm)
Short	-	100
Medium	100	120
Tall	120	-

Quality Control Program

- Solution:

```c
#include <stdio.h>
void main() {
    float widthM = 0.21;
    float heightM = 0.15;

    float widthMM = widthM * 1000.0;
    float heightMM = heightM * 1000.0;

    if (widthMM < 200 || widthMM > 230) {
        printf("width %0.2f mm is unacceptable \n", widthMM);
    }
    else {
        printf("width %0.2f mm is ok \n", widthMM);
    }

    if (widthMM >= 200 && widthMM <= 230) {
        printf("width %0.2f mm is ok \n", widthMM);
    }
    else {
        printf("width %0.2f mm is unacceptable \n", widthMM);
    }

    if (heightMM < 100) {
        printf("short\n");
    }
    else if (heightMM < 120) {
        printf("medium\n");
    }
    else {
        printf("tall\n");
    }

}
```