

Programming

CT103
Week 2b

Lecture Content

- Last lecture (Week 2a):
 - C basic variable types and their size in bytes.
 - Naming variables.
 - Declaring and initialising variables.
 - Comments.
- This evenings lecture (Week 2b):
 - Basic maths operators.
 - Modulus.
 - Else if statements.
 - Nested if statements.

MATH OPERATORS

Math Operators

- Addition: $+$
- Subtraction: $-$
- Multiplication: $*$
- Division: $/$

- Modulus (same as 'remainder' in maths): $\%$
 - This one is very useful!

Modulus

- Why is modulus % useful?
- The modulus operator allows us to get the remainder when doing integer division.
- What is the remainder?
 - When dividing two numbers that don't divide evenly, the remainder is what is left over.
 - E.g. $9/4 = 2$ with a remainder of 1.
 - In C: $9\%4 = 1$.

Modulus

- I still don't understand why is modulus % useful?
- Lets say I want you to write a program that tells me if a number is even or odd.
- How would you do it?

Odd or Even?

- Lets say I want you to write a program that tells me if a number is even or odd. How would you do it?
 - Answer: Use Modulus!

```
#include <stdio.h>
void main() {
    int num;
    printf("Enter a number:");
    scanf_s("%d", &num);
    if (num%2==0) {
        printf("Even");
    }
    else {
        printf("Odd");
    }
}
```

Odd or Even C Program

```
#include <stdio.h>
void main() {
    int num;
    printf("Enter a number:");
    scanf_s("%d", &num);
    if (num%2==0) {
        printf("Even");
    }
    else {
        printf("Odd");
    }
}
```

```
Enter a number:1
Odd
```

```
Enter a number:4
Even
```


ORDER OF OPERATORS

Order of operators

- C doesn't always compute maths operations in the order you might expect
- For example, is `ans` = 21 or 11?
`ans = 5 + 2 * 3;`
- C always does the multiplication before the addition

To be sure to be sure

$avg = i + j + k + l / 4;$

- C computes the division first, which means that

$avg = 10 + 2 + 4 + 8/4;$

Would be equal to 18.... Not what we want

- Always use parentheses, like:

$avg = (10 + 2 + 4 + 8)/4;$


Equal to 6

- In effect you are dictating explicitly the order you want operations evaluated in – much safer!

CHECKING IF NUMBERS ARE EQUAL

= or ==

- You might have noticed in the previous example of modulus that we used == in our IF statement.
- Why did you do this? Was this a typo?



```
if (num%2==0) {
```

= Or ==

- **Assigning value:** A single equals sign (=) assigns a value to something (e.g. `int i = 5;`)
 - Used when initializing or setting variables.
- **Checking equality:** Two equals signs together (==) is a relational operator to check what is on either side of the operator is the same
 - Often used in IF statements and While loops.
 - The result is either `true` or `false`
 - In C, true is 1, and false is 0

Equality Example

- **Checking equality:**

```
if (num==1) {  
    printf("number is 1 \n");  
}  
else {  
    printf("number is NOT 1 \n");  
}
```

```
Enter a number:1  
number is 1
```

```
Enter a number:55  
number is NOT 1
```

Testing Data

- The *if* statement works like:
 - If something is true then do A, otherwise do B
- C's Relational Operators:

Relational Operator	Description
==	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
!=	Not equal to

Examples

```
int i= 5;  
int j = 10;  
int k = 15;  
int l = 5;
```

- The following are **true**

i == l j < k k > l j != k

- The following are **false**

i > j k < j k == l

True vs False

- In standard C there is no Boolean data type, so we usually use an integer to store a true / false value
- It is very easy, because in C
 - True is represented by 1
 - False is represented by 0

Example – try it out yourself

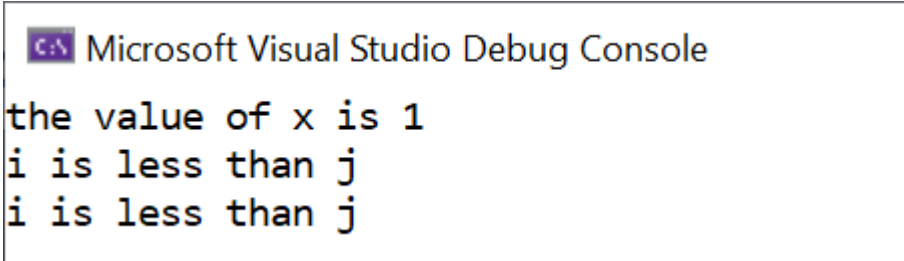
```
int i = 4, j = 7;
```

```
int x = (i < j);
```

```
printf("the value of x is %d \n", x);
```

```
if (x)
{
    printf("i is less than j \n");
}
```

```
if (i < j)
{
    printf("i is less than j \n");
}
```



C:\ Microsoft Visual Studio Debug Console

```
the value of x is 1
i is less than j
i is less than j
```

So...

- What is in a and b after these lines are executed?:

```
int a,b;  
a = (4 < 10);  
b = (8 == 9);
```

- Put them in a program and see for yourself if you are not sure

IF, ELSE IF, ELSE

More than one Decision

- Up until now, we have only considered a simple if – else statement,
 - i.e. `if (True){// do something} else{//do something different}`
- What do we do if we have multiple conditions?
 - If the grade > 85, A. If grade > 70, B. If grade > 55, C... etc.
- Will simply using multiple if statements work?

Grade Example

- Will the following code work as we want?

```
int grade = 81;
if (grade>85) {
    printf("A \n");
}
if (grade > 70){
    printf("B \n");
}
if (grade > 55) {
    printf("C \n");
}
if (grade > 40) {
    printf("D \n");
}
else {
    printf("F \n");
}
```

Grade Example

- Will the following code work as we want?

- Output:

B
C
D


- The student got a B. Why is the program also printing C and D?

```
int grade = 81;
if (grade > 85) {
    printf("A \n");
}
if (grade > 70) {
    printf("B \n");
}
if (grade > 55) {
    printf("C \n");
}
if (grade > 40) {
    printf("D \n");
}
else {
    printf("F \n");
}
```


Else if

- We need to use 'Else if' statements!
- Will the new code now work as we want?

```
int grade = 81;
if (grade > 85) {
    printf("A \n");
}
else if (grade > 70) {
    printf("B \n");
}
else if (grade > 55) {
    printf("C \n");
}
else if (grade > 40) {
    printf("D \n");
}
else {
    printf("F \n");
}
```



Else if

- Will the new code now work as we want?
- Output: **B**
- Success!
- ‘Else if’ will not check subsequent ‘If’ statements after a condition is **True**.

```
int grade = 81;
if (grade > 85) {
    printf("A \n");
}
else if (grade > 70) {
    printf("B \n");
}
else if (grade > 55) {
    printf("C \n");
}
else if (grade > 40) {
    printf("D \n");
}
else {
    printf("F \n");
}
```

CHECKING TWO CONDITIONS

Two Conditions in an If Statement

- Up until now, we have only considered a single condition in our if statement.
- What if we want to check if two conditions are true?
- For example:
 - If there is no rain and it is warm, bring sunscreen.
- How would we write a program to do this?

Two Conditions in an If Statement

- There are two ways of doing this.
- The first method is to use one if statement within another if statement. These are called '**nested**' if statements.

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

bring suncream

Two Conditions in an If Statement

- The second way to do this is to use **Boolean logic**.
- This involves using **AND**, represented by **&&** in C.
- This makes our code shorter.
- We will discuss Boolean logic next Monday in more detail.

```
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");
}
```

bring suncream