Chapter 2 – Conditional Statements

By the end of today's lesson you should be able to:

- Use IF and CASE statements in your programs
- Choose appropriate & efficient conditional statements and structures to use in your code

Key definitions:

- Condition
- Logical / Boolean operator
- Relational operator
- Comparison/ conditional statement
- Nesting

What is a conditional statement?

IFs and CASE statements are conditional lines of code – i.e. a certain pathway through code will be taken depending on the condition(s) of a variable or set of values.

Sometimes there may only be one condition, sometimes there may be several e.g.

- 'IF snooze = pressed, I'll be late for College, otherwise I won't'.
- 'IF I get >70 I'll get a B, IF I get >80 I'll get an A, otherwise I'll re-sit'.

We also often use these for menu type systems when we want a person to choose from a set of options, each of which might jump to a particular section of code/ a different subroutine.

Relational Operators

The following are the **relational operators** for use with IF statements (and many others you will use in C).

These compare two or more values against each other e.g.

Operator	Meaning
x < y	True IF x is *less than* y
x > y	True IF x is *greater than* y
x <= y	True IF x is *less than or equal to* y
x >= y	True IF x is *greater than or equal to* y
x != y	True IF x is *not equal* to y
x == y	True IF x is *equal to* y

Conditional Rules

- When using the equals operator make sure you use two equals signs (==) or the compiler will think you are creating an assignment statement for a variable e.g.:
  - Using two equals signs is a **comparison/ conditional statement** e.g. **if (age == 18)** this means you're using equals as a **relational operator**
  - If we used **age = 18** this would be an **assignment statement** and would store the value 18 in the variable age

Structuring IFs in C

```
//simple IF statement
if ( numStudents > 22)    {
        printf("\n Class is full. \n");
}

//simple IF ELSE structure
if (gcseMaths < 6)  {
            printf("You cannot take Comp Sci AL ");
}
else {
            printf("You're in!");
}
```

- The conditional part of the IF statement needs to be in brackets e.g. (age>17)
- A conditional statement **DOESN'T have a semi-colon** at the end; not on IF, ELSE IF, ELSE
- **Indent** the statements – this is **good practice** so you can see what's a part of the IF. Unlike in Python, the indents don't control when it stops!
- The curly brackets { } are used around control structures in C to define what is carried out inside them. Most IDEs have **bracket-matching** turned on to try to help you match them up.

Logical Operators

There are also three logical operators you can use to combine tests: AND, OR and NOT.

The correct C syntax for these is:

AND	test1 **&&** test2	True only if test1 AND test2 are true
OR	test1 **||** test2	True if test one OR two are true (including both being true)
NOT	**!** test1	True only if test1 is false (NOT)

You can chain these conditions together to make more complex conditional tests e.g.

```
if ( num>0  &&  num <=5 )        //If num is between 1 and 5
if (num ==5 || num ==0)          //IF num is 5 or num is 0
```

Exercise 1

Write a simple guessing game program (NO LOOPs at this point!).

Requirements:

- Create a variable that holds a randomly generated number between 1 and 10 (link for how to do this in C on the VLE C Chapters topic)
- If the user types in the right number as their guess the program should output "Congratulations"
- If the guess was within one number either side of the correct value, the program should output "So close"
- Otherwise, the program should generate "Sorry wrong answer"

Nested IFs

- 'Nesting' = when one control statement is inside another e.g. an IF inside an IF or an IF inside a loop
- Sometimes it can be more efficient to nest your IFs and loops so that certain statements are dependent on another running e.g. we only want a path in a program to be followed IF a previous decision is true
- IF a student is 15 or over
  - Only then check IF student has got at least six GCSEs at grade C/ 4 or above (e.g. meets the BHASVIC entry reqs):
    - And only then IF they have at least a 6/ B in Maths they can enrol on the Computer Science A Level course

Exercise 2

Write a program for a tourist attraction that calculates and outputs the total charge for a group.

It should:

- Allow a user to enter W, S or B character for day of the week (Weekday, Sat/Sun or Bank Holiday)
- Enter how many Child tickets and how many Adult tickets they require
- Child tickets are £5 on weekdays or £7.50 on a weekend and a bank holiday
- Adult tickets cost £8 on weekdays, £12 on weekends and £14 on bank holidays
- If there are more than 8 people in the group, a 10% discount is applied
- The program should output the total costs of each ticket type, as well as the overall total, showing any discount that has been applied
- Your program should be as efficient as possible in terms of the IF structures you use

CASE statements

```
int main()
{
  char user_choice;
  printf("Please choose either E, A or D:");
  scanf("%c",&user_choice);

  switch (user_choice)
  {
        case 'E':
              printf("You have chosen Elephant\n");
              break;
        case 'A':
              printf("You have chosen Aardvark\n");
              break;
        case 'D':
              printf("You have chosen Dolphin\n");
              break;
        default:
              printf("Choice was %c \n", user_choice);
              printf("Please try again with E, A or D \n");
              break;
  }

return 0;
}
```

- The variable **user_choice** is passed into the SWITCH function
- All the CASE statements are searched to see if there's an exact match with the variable.
- Each CASE ends with a normal colon **:**
- If there is a match, the line of code within that CASE will be executed, otherwise they are ignored.
- You can have many different things that happen inside each CASE
- Default is like the else part of an IF – it will be carried out when no other CASE has been met. You don't have to include one.
- When **break** is reached the program will jump out of the case statement and continue the rest of the program.
- Notice that there are curly brackets containing the switch code.

Exercise 3

Write a program that uses a CASE structure to act as a simple calculator.

Requirements:

- Get 2 numbers as user inputs
- Ask the user if they would like to add, subtract, multiply or divide (+, -, *, /)
- Use SWITCH CASE statements to do the different operations and carry out the calculations (no subroutines yet please!)
- Output both the sum and the result to the user **e.g. 15 / 3 = 5**
- Add **validation** to ensure that data entered is numbers only
- Be careful with data types used here, make sure you test your program with both integer and float numbers

Exercise 4

A simple classification system asks a series of Yes/No questions in order to work out what type of animal someone is looking at e.g. Does it have four legs? Does it eat meat? Is it stripy?

Write a program to help the user decide between the following animals:
Horse, cow, pig, sheep, dog, cat, lion, tiger, whale, dolphin, seal, penguin, ostrich, sparrow, spider, ant, bee, wasp, termite, octopus, squid

You should NOT solve this by writing 101 IF statements. Think about efficient ways to group your data! What is the fewest Qs you can use?

You're also not allowed to ask questions like "Is it a cat?"
……**that's cheating**!

Validation Functions in C

- There are lots of in-built C functions that we can use to try and stop people from entering crap into our programs, or that can help to convert it to useful data.

  ***** To use these functions you must add #include <ctype.h> at the top of your program. *****

- **toupper()** converts a character to an upper case value
- **tolower()** converts a character to a lower case value

- **isalnum()** returns a non zero value if alphanumeric A-Z, a-z or 0-9
- **isalpha()** returns a non zero value if a letter A-Z or a-z
- **isdigit()** returns a non zero value if decimal digit
- **islower()** returns a non zero value if an lower case letter a-z
- **isupper()** returns a non zero value if an upper case letter A-Z
- **ispunct()** returns a non zero value if a punctuation character
- **isspace()** returns a non zero value if whitespace i.e. spaces, tabs etc.

- Some of these can behave a bit strangely - they won't always be able to be used, as sometimes it depends on the data type you are storing your values as e.g. if you type a letter into an int variable, it will be stored as a 0 (it just didn't read in an integer) so isalpha() may not be useful – this is where the error checking you saw in Chapter 1 might help!

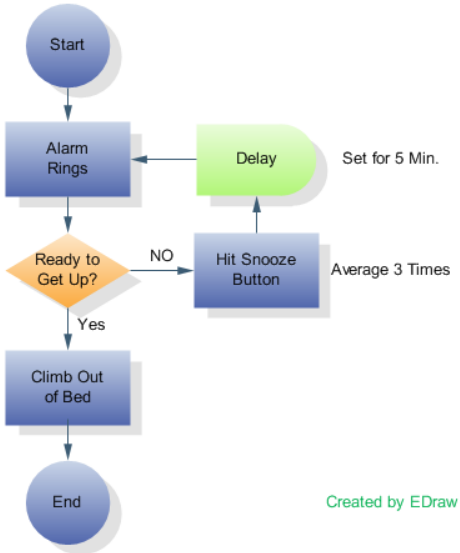
Run this program & test it:

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

int main(){
        char gender;
        int age=0;
        printf("Input gender (M or F) and then your age");
        scanf("%c%d", &gender, &age);
        if (toupper(gender) == 'F'){
                printf("You chose Female and you entered %c", gender);}
        else if (toupper(gender)=='M'){
                printf("You chose Male and you entered %c", gender);
        }
        if (islower(gender) != 0){
                printf("\n You entered a lowercase letter");}
        else if (isupper(gender)!= 0 ){
                printf("\n You entered an uppercase letter");}
        else if (isalpha(gender)==0){
                 printf("You didn't enter a letter");}
        else if (ispunct(gender) !=0){
                printf("\n You entered punctuation");}
        else if (isspace(gender) !=0){
                printf("\n You entered whitespace");
        }
        if (isdigit(age) == 0){
                printf("You didn't enter a number for age");
        }
        system("pause");
        return 0;
}
```

1. Run the program a couple of times with different inputs:
   - make sure that you understand how the different validation functions work (and if they do at all!)
   - Try entering valid and invalid data and look at which statements are output

Decisions & flowcharts



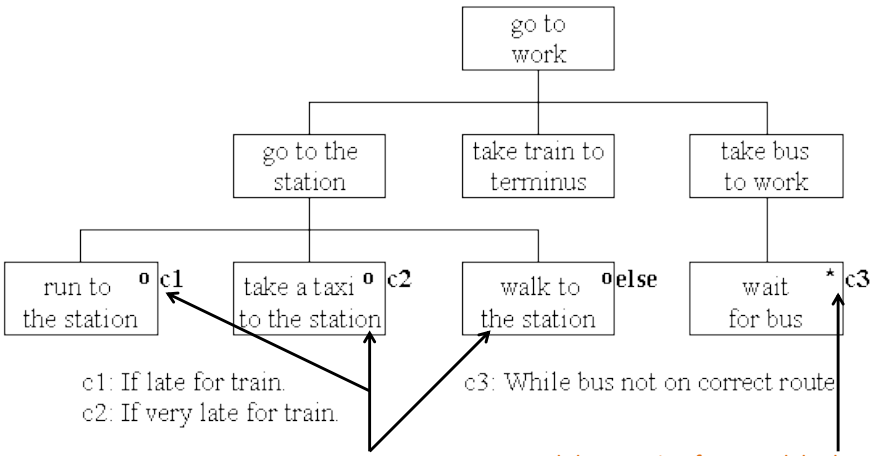
Identify decision points

A floor cleaning vacuum robot is in a room, the room has 4 walls and several obstacles positioned around the room. It moves forward, backwards and can rotate at a given angle. The robot has sensors to help it carry out its task.

Produce a flow chart in Visio/ Draw.io for the robot so that it can move around and clean as much of the available floor space as possible.

- When do decisions need to be made?
- What are the conditions that affect these decisions?
- How do the decisions affect the flow through the program?

Structure diagrams revisited



See that the notation for a module that is CONDITIONAL (IF) has a circle in the corner

And the notation for a module that is ITERATIVE (loops) has an asterisk