COMP1511 PROGRAMMING FUNDAMENTALS

Lecture 4

Loop the loop

YESTERDAY...

- More complex IF statements
- Logical operators
- Chaining IF and ELSE
- Breaking stuff

- Structs

TODAY...

- Looping

WHERE IS THE CODE?

LIVE LECTURE CODE CAN BE FOUND HERE:

https://cgi.cse.unsw.edu.au/~cs1511/21T3/live/Week02/

WHEN DO WE NEED TO LOOP?

REPETITION

- Any time your program needs to *keep doing something (repeating the same or similar action)* until something happens and you may not know how many times that will be in advance

- Can you think of some examples in real life?
  - While there are songs in my playlist, keep playing the songs

WHILE

REPETITIVE TASKS SHOULDN'T REQUIRE REPETITIVE CODING

- C normally executes in order, line by line (starting with the main function after any # commands have been executed)
- if statements allow us to "turn on or off" parts of our code
- But up until now, we don't have a way to repeat code
- Copy-pasting the same code again and again is not a feasible solution
- Let's see an example where it is inefficient to copy and paste code...

WHILE

WHILE SOMETHING IS TRUE, DO SOMETHING

while loops

- Count loops
- Sentinel loops
- Conditional loops

```
// expression is checked at the start of every loop
while (expression) {
    // this will run again and again
    // until the expression is evaluated as false
}
// When the program reaches this }, it will jump
// back to the start of the while loop
```

CONTROL THE WHILE LOOP

```
// 1. Initialise the loop control variable before the loop starts

while (expression) { // 2. Test the loop control variable, done within the
                        (expression)

    // this will run again and again
    // until the expression is evaluated as false

    // 3. Update the loop control variable - usually done as the last
    statement in the while loop
}
// When the program reaches this }, it will jump
// back to the start of the while loop
```

TO INFINITY AND BEYOND

TERMINATING YOUR LOOP

- It's actually very easy to make a program that goes forever
- Consider the following while loop:

```
while (1 < 2) {
    // Never going to give you up
    // Never going to let you down . . .
}
```

CONTROL THE WHILE LOOP

COUNT LOOPS

- Use a variable to control how many times a loop runs - a "loop counter"
- It's an **int** that's declared outside the loop
- It's "termination condition" can be checked in the while expression
- It will be updated inside the loop

```
//1. Declare and initialise loop control variable (just outside while loop)
int count = 0;

while (count < 5) { //2. Test the loop control
                    //   variable against counter
    printf("What is a score you want to enter? ");
    scanf("%d", &score);
    sum = sum + score;
    printf("You have gone around the loop %d times, and the sum is %d now\n", count, sum);

    count = count + 1; //3. Update the loop control variable
}
```

SENTINEL VALUES

WHAT IS A SENTINEL?

- When we use a loop counter, we assume that we know how many times we need to repeat something
- Consider a situation where you don't know the number of repetitions required, but you need to repeat whilst there is valid data
- A sentinel value is a 'flag value', it tells the loop when it can stop…
- For example, keep scanning in numbers until an odd number is encountered
  - We do not know how many numbers we will have to scan before this happens
  - We know that we can stop when we see an odd number

CONTROL THE WHILE LOOP

SENTINEL LOOPS

```
//1. Initialise the loop control variable
int end_loop = 0;

while (end_loop == 0) { //2. Test the loop condition

    printf("Please enter a score to add to the sum: ");
    scanf("%d", &score);
    if (score > 0) {
        if (score % 2 == 0) {
            sum = sum + score;
        }
    } else {
        //3. Update the loop control variable
        end_loop = 1;
    }
}
```

- Sentinel Loops: can also use a variable to decide to exit a loop at any time
- We call this variable a "sentinel"
- It's like an on/off switch for the loop
- It is declared and set outside the loop
- It's "termination condition" can be checked in the while expression
- It will be updated inside the loop (often attached to a decision statement)

CONTROL THE WHILE LOOP

CONDITIONAL LOOPS

- Conditional Loops: can also use a condition to decide to exit a loop at any time
- This is called conditional looping
- Also do not know how many times we may need to repeat.
- We will terminate as a result of some type of calculation

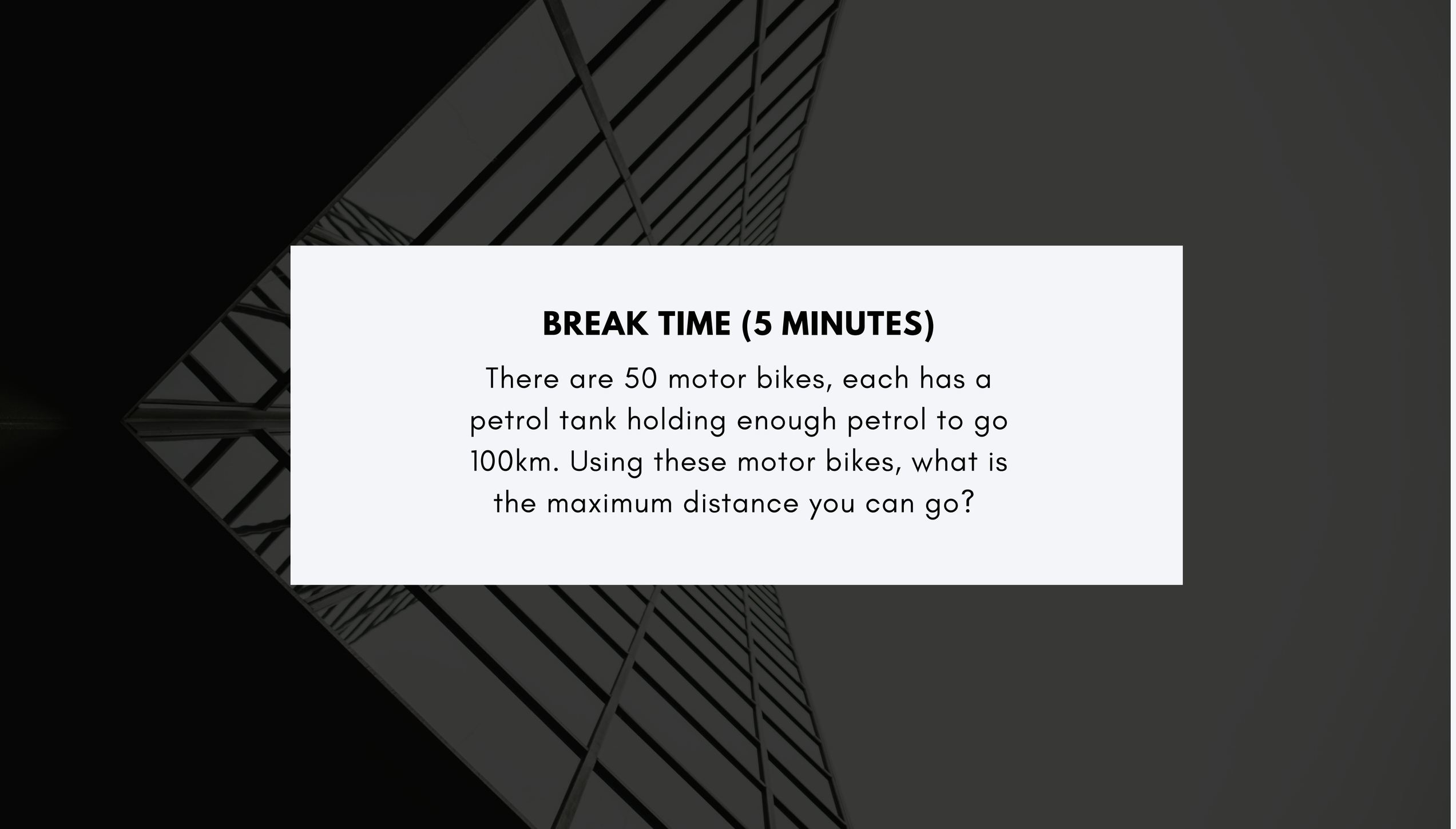
```
//1. Initialise the loop control variable
// Since I want the sum to be as close to 100, that is my control condition
int sum = 0;

while (sum < 100) { //2. Test the loop condition

    printf("Please enter a score to add to the sum: ");
    scanf("%d", &score);

    //3. Update the loop control variable
    sum = sum + score;
}
```

LET'S SEE IT IN ACTION

CODE DEMO TIME

https://cgi.cse.unsw.edu.au/~cs1511/21T3/live/Week02

- While loop with a counter
  while_count.c
- While loop with a sentinel value
  while_sentinel.c
- While loop with a condition
  while_condition.c



BREAK TIME (5 MINUTES)

There are 50 motor bikes, each has a petrol tank holding enough petrol to go 100km. Using these motor bikes, what is the maximum distance you can go?

WHILE INSIDE A WHILE

PUTTING A LOOP INSIDE A LOOP

If we put a loop inside a loop . . .

- Each time a loop runs
- It runs the other loop
- The inside loop ends up running a LOT of times

PROBLEM TIME

1. **PRINT OUT A GRID OF NUMBERS**

Print out a grid of numbers:

1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5

1. Break down the problem...
2. Get it down to a component that you can do...

PROBLEM TIME

2.PRINT OUT A PYRAMID OF NUMBERS

What if we now print out a half pyramid of numbers:

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

1. Break down the problem...
2. Get it down to a component that you can do...

WHAT DID WE LEARN TODAY?

LOOP THE LOOP WHILE (COUNTER)

while_counter.c

LOOP THE LOOP WHILE (SENTINEL)

while_sentinel.c

LOOP THE LOOP WHILE (CONDITION)

while_condition.c

LOOP INSIDE A LOOP

grid: print_grid.c

pyramid: print_pyramid.c

ANY QUESTIONS?

DON'T FORGET YOU CAN ALWAYS EMAIL US ON CS1511@CSE.UNSW.EDU.AU FOR ANY ADMIN QUESTIONS

PLEASE ASK IN THE FORUM FOR CONTENT RELATED QUESTIONS