**Purpose**

The purpose of unit 04 was to introduce us to loops and how to read and write files

**Concepts**

**Loop** – A repetitive task that runs until a statement forces the loop execution to end. This could be anything from a comparing numbers, bools, strings, or array sizes, etc... Loops are essential for programmers since they allow us to reuse code and perform the same task as many times as we desire without having to rewrite the same task repeatedly.

**Counted loop** – Often a for loop or a loop that ends when a predefined iterated number has met a maximum or minimum threshold. The classic way to do this is using a for loop.

for($i = 0; $i < THRESHOLD\_MAX; $i++)

The above code would run whatever was inside of its closure until THRESHOLD\_MAX is reached and automatically iterates $i at the end of the closure’s execution. The same can be applied manually in a while loop.

**Infinite loop** - A loop that continues execution forever and never breaks out of its closure. These loops are dangerous since they eat system resources like memory, CPU cycles, and even hard disk space. Most modern programming languages will not allow infinite loops to consume an entire systems recourse, but you can still do this quite easily in C.

**Pre-Test loop** – A loop where the breaking condition is tested at the beginning of the loop. This is how while loops work

While(true) {

}

This says that while true is true execute this code block. This is also an example of an infinite loop.

**Post-Test Loop** – These loops check the break value after a codes execution, it is common to use counted loops like for loops in this scenario.

for($i = 0; $i < THRESHOLD\_MAX; $i++)

After executing the code in this loops closure, the value of $i will be tested against the value of THRESHOLD\_MAX if that value ever becomes greater but not equal to the THRESHOLD\_MAX then the loop will break on not continue executing its closure

**Implications**

Loops are an excellent way to describe the inherent power of programming. Since all programming is meant to find a solution to one problem or another, programs should also be as efficient and precise as possible to come to accurate conclusions as quickly as possible.

Using loops makes programs more efficient and uses less space than if you were to write each execution by hand. Of course, you are trading memory for CPU cycles, but more importantly you are trading human time for computer time. The more that your program can effectively use loops, the less time programmers have to spend writing code and wasting their efforts on repetitive tasks.