Create a tutorial for a new programmer.  This tutorial should help the programmer understand the difference between while loops and for loops.  Be sure to provide supportive examples that include syntactically correct code.

Once you have made your post, please respond to at least two other students interacting on the tutorial posted and differences between while loops and for loops.

Hi everyone,

Today we’re going to discuss loops, and specifically hone in on two common types of loops. But first, let’s define what a loop is and how/why we might use them.

**LOOP:**  
A loop is a structure that repeatedly executes **its body of instructions** as long as the associated **condition evaluates to true**. When the condition evaluates to false, control exits the loop. Each execution of the loop body is referred to as an **iteration**.

A real life example is something like the following:

Consider a prompt like “The kids can watch television after dinner and until their bedtime, at 8 PM”. The dinner time varies from night to night, but you always check the clock at half hour intervals to see if it is 8PM.

On Monday, dinner ended at 7pm. So you check the clock at 7:30 – it’s not yet 8, so they can continue watching television. You check again at 8PM, and the kids stop watching TV and go to sleep.   
Maybe Tuesday, you finish dinner earlier at 6:30 PM.

You continually check the clock each half hour until it reaches 8 PM and put the kids to bed.

The body of instructions is that we allow the kids to continue watching TV. The condition is that the time is before 8 PM. Once the time is 8PM or later, a different instruction (putting the kids to bed) can be executed. Notice in this example that we check the clock each half hour. But because dinner ends at different times each night, we do not know necessarily know how many times we check the clock (assuming we do not do the math), thus we don’t know the number of iterations.

**WHILE LOOPS:**

A while loop runs a block of code repeatedly while a certain condition is met. Once the loop starts an iteration, it finishes that cycle completely, even if the condition changes to false in the middle.

The example above could be programmed into a while loop.

Let’s say that rather than the parents checking the time periodically, we program an alarm to do the work.

This code would look like this:

#include <iostream>

using namespace std;

int main() {

double currentTime; // time in hours, e.g., 6.5 for 6:30 PM

const double bedTime = 8.0;

cout << "Enter the time dinner ended (e.g., 6 for 6 PM, 6.5 for 6:30 PM): ";

cin >> currentTime;

cout << "The kids can now watch TV!\n";

// Check every half hour until it's 8 PM

while (currentTime < bedTime) {

currentTime += 0.5; // add 30 minutes

cout << "Checking the time... it's " << currentTime << " PM." << endl;

if (currentTime < bedTime) {

cout << "Not bedtime yet — they can keep watching TV.\n";

} else {

cout << "It's 8 PM! Time for bed.\n";

}

}

return 0;

**FOR LOOPS:**

Another type of loop that is commonly used are FOR loops. A for loop iterates through code for a **set number of times**. It’s handy when you know exactly how many repetitions you need, like counting from 1 to 10 or going through items in a list.

Let’s change the example above. Now the family finishes dinner each night at 6pm, but the kids have homework readings they need to finish before getting to watch TV before bed. Each night, they must read 5 pages.

Therefore, we know that all we need to do is check with the user how many pages the kids have read. An example might look like this:

#include <iostream>

using namespace std;

int main() {

const int totalPages = 5;

int minutesPerPage;

cout << "Enter how many minutes it takes to read one page: ";

cin >> minutesPerPage;

int totalMinutes = 0;

for (int i = 0; i < totalPages; i++) {

totalMinutes += minutesPerPage;

}

int hours = totalMinutes / 60;

int minutes = totalMinutes % 60;

cout << "\nDinner ended at 6:00 PM.\n";

cout << "The kids can start watching TV at approximately "

<< (6 + hours) << ":";

if (minutes < 10) cout << "0"; // keeps formatting nice (e.g., 6:05)

cout << minutes << " PM.\n";

return 0;

}

Here, the code will iterate 5 times. If the kids have 10 pages to read, then the iteration number is known, and the code will iterate 10 times.