Infinite Loops

This lesson explains how infinite loops might arise in a while loop using an example

We'll cover the following

Arisal of Infinite Loops

One common programming mistake is to create an **infinite** loop. An **infinite** loop refers to a loop, which under certain valid (or at least plausible) input, will never **exit**.

> **Note:** Beginning programmers should be careful to examine all the *possible* inputs into a *loop* to ensure that for each such set of inputs, there is an **exit** condition that will eventually be reached.

Compilers, *debuggers*, and other *programming tools* can only help the programmer so far in detecting **infinite** loops.

> **Note:** In the fully general case, it is not possible to automatically detect an infinite loop. This is known as the **halting** problem.

While the halting problem is not solvable in the fully general case, it is possible to determine whether a loop will **halt** for some *specific* cases.

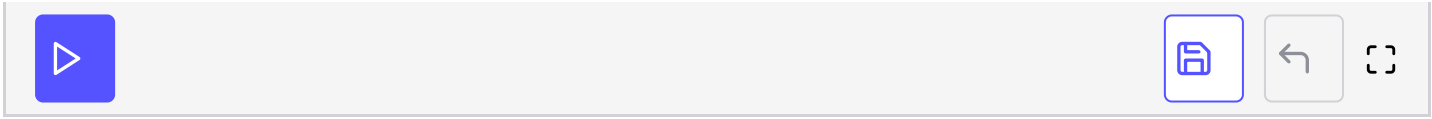
Example of Infinite loop

Down below is an example of an *infinite* loop.

> **Note:** You will get an **error** when you try to run the code below due to limitations of our platform.

limitations of our platform.

```
#include <iostream>
using namespace std;
int main()
{
  int a = 1;
    while(a) { // the while condition will always be met as will always return true
      cout << "Infinite loop\n" << endl;
    }
    return 0;
}
```



In a normal case, the code above will print **“Infinite loop”** without stopping because the condition statement in the `while` loop will always resolve to **true**. There is no point at which it’ll return **false** hence we’ll get stuck in an *infinite* loop, and the code will execute forever.

Well this marks the end of the chapter on **loops**. Next up, we’ll discuss *functions* in C++.