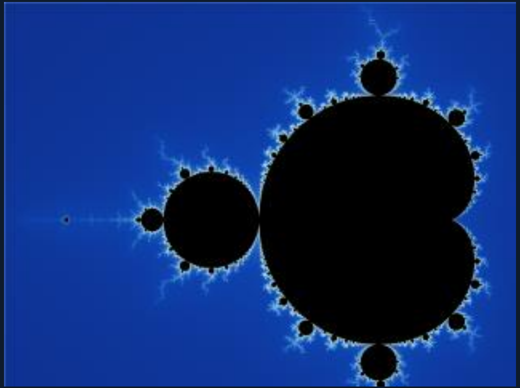
Loops



Contents

Loops

A loop is similar to an if statement, however the statement is repeatedly executed as long as the conditional expression is true.

Many, many algorithms require repetition

- Without loops, we would need to type the same thing out again and again and again and...

Loops & Algorithms

- Suppose we wish to find the location of the number 100 in the following table.

Location	1	2	3	4	5	6	7	8
Number	11	22	345	85	234	199	100	1

- We can easily see that this number is at location 7, but a computer must search the list one location at a time.

C++ has a few types of loops available

- While
- Do-while
- For
- For each

While Loops

- Look like...

```
while(condition)    //there is no ;
{
    //Code goes here
}
```

- The body of the loop is usually a code block surrounded by braces, but it could be a single statement.

Main Game Loop

- Most games will make use of a while loop

```
while(gameBeingPlayed)
{
    /* respond to user events
    process keyboard input
    process mouse input
    update game status
    render a frame */
}
```

Do while

- Very similar to the while loop, but this time the conditional evaluation is at the end of the loop.
- This means that the body of the loop is always executed at least once.

```
do
{
    //Code goes here

} while(condition); //Condition is now here
```

For

- The for loop. You will use this one **a lot**.

```
for(initialiser; condition; expression)
{
    //Code goes here
}
```

- Has three parts
  - Initialiser: Executed once, before the loop begins.
  - Condition: Evaluated before each execution of the body
  - Expression: Is executed after the body

For loop Examples

Take a guess what is printed for each of the following:

```cpp
int numberOfIterations = 10;
for(int i = 0; i < numberOfIterations; i++)
{
    std::cout << i << std::endl;
}
```

```cpp
for(int i = 0; i <= 10; i++)
{
    std::cout << i << std::endl;
}
```

```cpp
for(int i = 10; i > 0; i--)
{
    std::cout << i << std::endl;
}
```

```cpp
for(int i = 0; i > 0; i++)
{
    std::cout << i << std::endl;
}
```

But I’d like to get off now!

- Any loop can be exited at any point through the use of the break statement.

```
for(int i = 0; i < 10; i++)
{
    if(i > 4)
    {
        break; //Exits the loop
    }

    std::cout << i << std::endl;
}
```

- What is the output of the above code?

Continuing on...

- The continue statement is a loop control statement, much like the break statement.

- Instead of terminating the loop, it causes execution to recommence at the top of the loop.

```
for(int i = 0; i < 10; i++)
{
    if(i % 2 == 0)
    {
        continue;
        //go directly to the top, execute the expression and check the condition.
    }

    std::cout << i << std::endl;
}
```

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Summary

- Loops allow us to repeat blocks of code multiple times while the conditional is true.

- You can use any type of loop, however some loops are better suited to certain types of problems.
  - In particular, for loops work well with arrays!

- The break and continue statements allow us to have limited control of the loop.