Loops, Arrays, and Number Sequences

Topics: while-loop, do-while loop, for-loop, arrays
Lab 2: https://maryash.github.io/135/labs/lab_02.html

- Sadab Hafiz

Introducing the While-loop

For-loops allow us to loop things for a certain number of times. However, there are situations when we need to repeat a task indefinitely as long as a certain condition is met every repetition.

We don’t necessarily know how many times the task needs to be repeated but we know when it should repeat. In those cases, a while-loop is used:

```
while(some condition is met) {

      ... do something

}

// the while-loop will only end when the condition is not met anymore (or break)
```

Infinite loop and keyword `break`

A while-loop should always allow an exit. If the condition is met every time infinitely, the while-loop will keep looping forever. This is known as an infinite loop. It is possible to break out of a while-loop using the keyword `break`. For example:

```
int a = 0;                                          // variable a starts at 0

while (true) {                                      // while-loop condition is always met

      // this while-loop would be an infinite loop without a break conditional

      if (a == 10) {                                // if a is equal to 10

            break;                                  // break out of the while-loop

      }

      a += 1;                                       // increment a by 1

}

// This while-loop repeats 10 times. Sounds familiar?
```

while-loop vs for-loop

A while-loop can do what a for-loop does:

```
int i = 3;                  // start

while (i < 7) {             // stop

      cout << i << " ";

      i++;                  // step

}
```

```
// for (start, stop, step)

for (int i = 3; i < 7; i++){

      cout << i << " ";

}

// Output for both: 3 4 5 6
```

Although both loops create the same output, the for-loop reads better and it is preferred in scenarios like this. In both cases, the stop condition is a boolean that needs to be true to keep repeating the loop. If it is false, the loop ends.

Introducing the do-while loop

The code inside a while-loop only executes if the loop condition is met. There are scenarios when you want to execute the code at least once before checking whether or not the condition is met. In such cases, a do-while loop is handy:

```
// get the age from user at least once and keeps asking for age if entered age is negative

int age = 0;                // initialize age variable

do {

     cin >> age;            // get age from user input

} while (age < 0);          // repeat if the entered age is negative
```

The same program can be written using only a while-loop. However, it would require writing more lines of code as you have to get the input once before the while-loop.

Introducing Arrays

An `int` variable allows us to store a single integer value. What if we need to store multiple `int` values in a list? We have to use arrays:

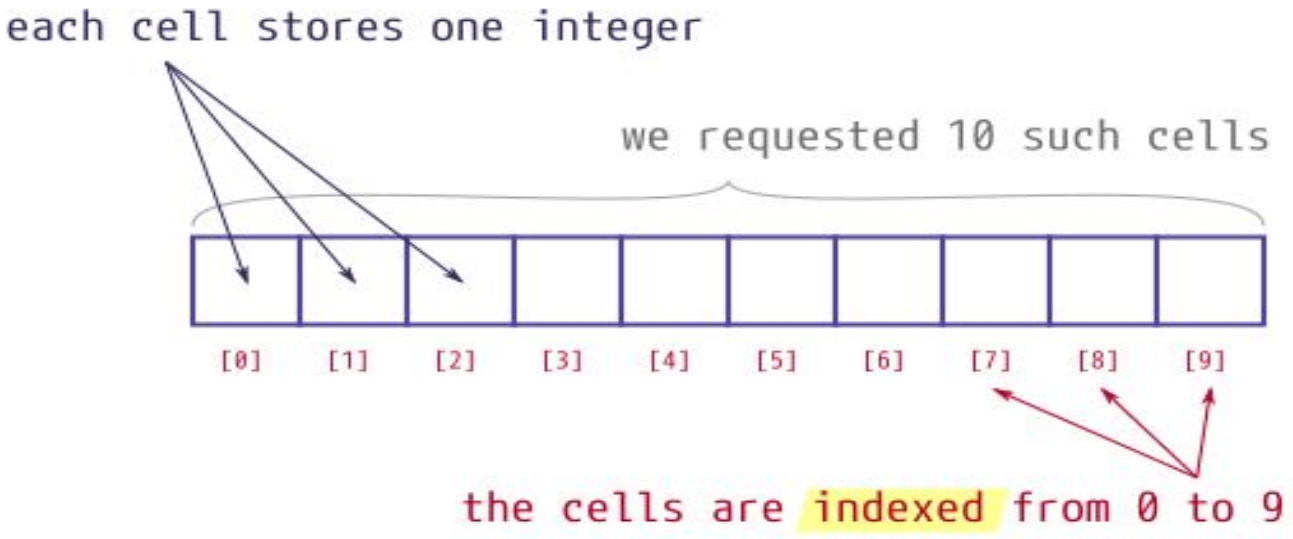
```
int myArray[3];                                    // create an empty integer array of size 3
myArray[0] = 12;                                   // set index 0 equal to 12
myArray[1] = 21;                                   // set index 1 equal to 21
myArray[2] = 2000;                                 // set index 2 equal to 2000
```

The same result can also be achieved using the following syntax:

```
int myArray[3] = { 12, 21, 2000 };                 // create an integer array containing 12,21,2000
```

Visualizing Arrays

```
int myData[10];                                    // empty integer array of size 10
```



Iterating through an Array using a for-loop

Items stored in an array can be accessed by their index. Using a for-loop it is possible to go through the entire array and access all items:

```
int myArray[3] = { 12, 21, 2000 };                  // create an integer array containing 12,21,2000

for (int i = 0; i < 3; i++) {                       // i will be 0,1,2

      cout << myArray[i] << " ";                    // print item in index i of myArray

}

// output: 12 21 2000
```

Iterating through an array can have other applications than just printing the array. For practice, can you add all the items in an array?