Array Initialization and Updating array elements

You will learn how to assign values to array elements after an array has been declared. You can also populate an array while it is initialized.

We'll cover the following

Assigning values to array elements

I can assign the integers 1 through 5 to my `grades` array like this:

```
#include <stdio.h>

int main ()
{
  int grades[5];
  int i;
  for (i=0; i<5; i++) {
    grades[i] = i+1;
  }
  for (i=0; i<5; i++) {
    printf("grades[%d]=%d\n", i, grades[i]);
  }
  return 0;
}
```

What can be **dangerous** however, is that I can also ask C to **assign** values to elements beyond the bounds of my array, and **C won't complain**:

```
#include <stdio.h>

int main ()
{
  int grades[5];
  int i;
  for (i=0; i<5; i++) {
    grades[i] = i+1;
  }
  grades[5] = 999;
```

```
  grades[500] = 12345;
  for (i=0; i<6; i++) {
    printf("grades[%d]=%d\n", i, grades[i]);

  }
  printf("grades[500]=%d\n", grades[500]);
  return 0;
}
```

The reason this is dangerous, is that **only 20 bytes of memory were set aside for the array** yet we are writing to locations in memory that are outside of these boundaries. The location in memory that is 24 bytes beyond the start of the `grades` array (the location accessed by the expression `grades[5]`), and the location in memory that is 2004 bytes beyond the start of the array (the location accessed by the expression `grades[500]`), have not been set aside for the `grades` array, and **these locations in memory may be used to store other important things**. They may correspond to some other variable in your program, or they may be storing some information that's not even part of your program at all, but is part of the operating system — for example some part of the OS that is controlling the hard disk, or the video screen, or the network, or the fans, etc. By **writing your data** in memory locations that are not within the bounds of memory that has been set aside by you, **anything can happen**.

One example: imagine you are writing a C program to control a robot arm (as we do in experiments in our lab). What do you think would happen if suddenly you over-wrote some random location in memory? Answer: nothing good!

Initializing Arrays

In general, we can initialize values of an array using curly brackets `{}` like this:

```
int grades[5] = {4, 3, 2, 5, 1};
```

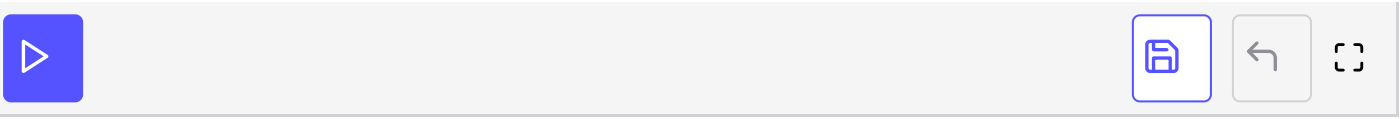
This says allocate the `grades` array to hold 5 values, and assign `4` to the first value, `3` to the second value, `2` to the third value, `5` to the fourth value, and `1` to the 5th value.

We can initialize specific values (and not others) like this:

```
#include <stdio.h>
```

```
int main ()
{
  int grades[5] = {[0]=1, [2]=3, [4]=5};

  int i;
  for (i=0; i<5; i++) {
    printf("grades[%d]=%d\n", i, grades[i]);
  }
  return 0;
}
```



This says initialize the 0th (the first) element of the array to contain the integer `1`, set the third element (`grades[2]`) to contain the integer `3`, and set the fifth element (`grades[4]`) to contain the integer `5`. **All other elements** are set to zero.

So up till this point, our arrays have worked in a single direction i.e. length. However, we can have arrays with multiple dimensions. We'll discuss this in the next lesson.