Variable-Length Arrays

You will be introduced to variable-length arrays so that you don't have to specify array size at compile time.

In the array examples before, we have hard-coded the size of the array. In modern C (the C99 standard and above) it is possible to declare arrays without knowing until run-time their size.

Here is how to do it:

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

int main (int argc, char *argv[])
{
  if (argc < 2) {
     printf("please provide an integer argument\n");

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

```
$ ./go 5
grades[0]=0
grades[1]=1
grades[2]=2
grades[3]=3
grades[4]=4
```

Output

The obvious benefit of allowing variable-length arrays, is that you don't have to know in advance of your program running, how much memory to allocate for your array variables. The downside of this, is that you have to guard against the possibility that your program will attempt to allocate too much memory.

If you notice in the code above, we’ve used the arguments which were passed into the program from the command-line. We’ll delve deep into this in the next lesson.