C Programming

Intro to Arrays

Example of NOT Using Arrays

float price2 = 5.99;

float price7 = 2.97;

float price1 = 4.99;

float price8 = 0.99;

float price3 = 2.79;

float price9 = 22.99;

float price11 = 1.19;

float price5 = 2.17;

float price6 = 4.19;

float price10 = 0.04;

float price4 = 4.29;

Nice? Nope!

This is workable with 3 prices but very awkward with 30 and impractical with 100.

There's a better way, though!

Yep, You Guessed It!

Arrays can be used to keep track of groups of data

Three conditions

1. All have the same data type

2. All deal with the same "topic"

3. There's a fixed maximum number of them

Creating an Array Variable for Prices

The array variable we can create to replace 30 distinct variables is:

float prices[30];

Data type

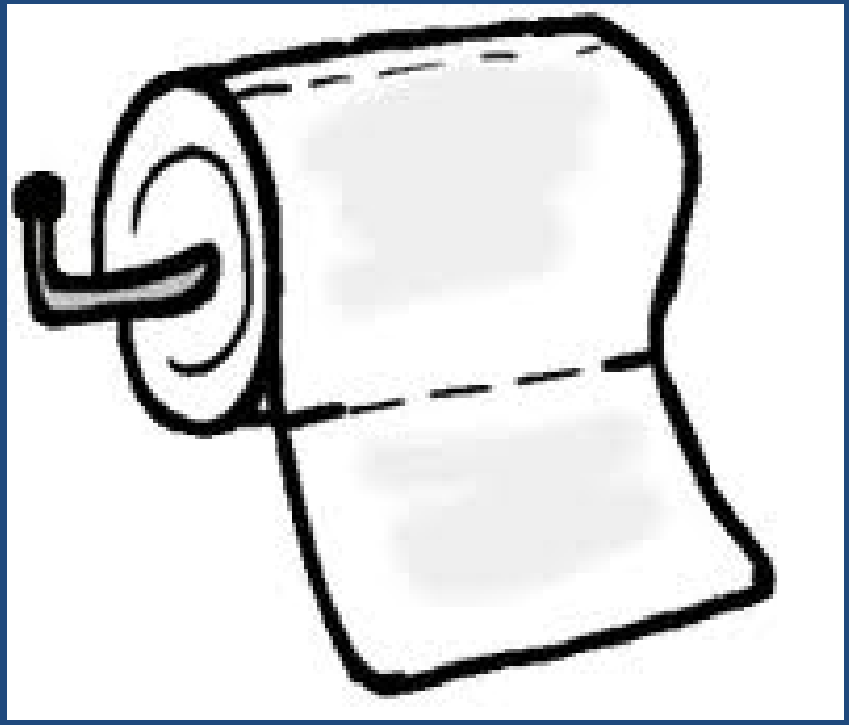
This is different from languages like Java and C#.

What is an array?

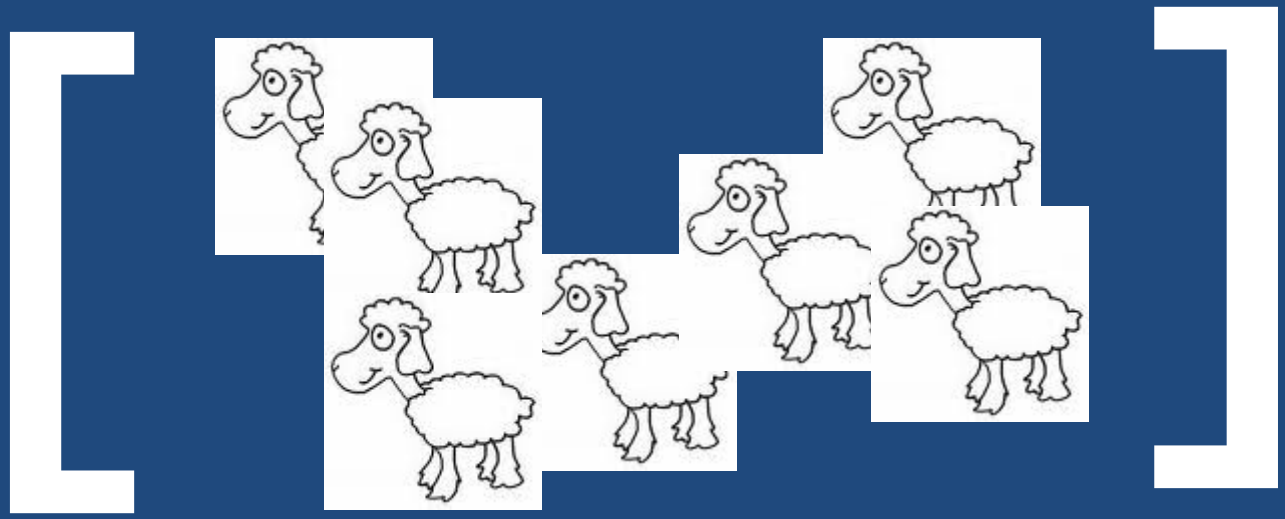
A variable

A group of elements of the same data type

The group is ONE variable with separate elements



The elements are dealt with using square brackets



Examples

All of the following are uninitialized array variable declarations:

- float prices[30];

- This is an array of 30 prices, each of which is an float.

```
int countryPopulation[200];
```

```
char name[81];
```

Examining the Syntax

float prices[30];

Syntax	What
float	data type of the elements of the array
prices	name of the variable
[ ]	square brackets surrounding the index
30	number of elements in the array
;	termination

Individual Data in the Array

Each piece of data stored within the array is called an **element.**

Use an **index** between the square brackets

```
printf("%f\n", prices[10]);
/* prints element #10 in the
prices array */
```

Array Indeces

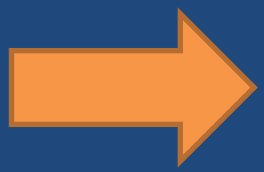
The first element of an array has index 0.

- e.g. printf("%f\n", prices[0]);

The last element of an array has an index equal to **one less** than the number of elements indicated in the declaration.

e.g. printf("%f\n", prices[29]);

If prices was declared to have 30 elements, indeces can go from 0 to 29



Too Many?

What if you forget that and try to access element #30?

- Result: garbage!

The compiler will not do any sanity checking on your index.

Implications of No Range Checking For Indeces

If prices is of size 30, all of the following are **legal**, if not necessarily **valid**:

- printf("%f\n", prices[30]);
- printf("%f\n", prices[400]);
- printf("%f\n", prices[-10]);
- printf("%f\n", prices[myVariable]);
- printf("%f\n", prices[doSomething()]);
- printf("%f\n", prices[40 + myVariable]);

Working With Array Elements

Use the name of the variable, followed by the index between square brackets.

- e.g. prices[29] = 23.99;
- e.g. printf("%f\n", prices[3]);
- e.g. prices[29] = prices[1] * 1.19;

You can treat an array element just like a normal variable when you're using it.

Uninitialized Array Elements

So far, all of the variable declarations we've had have not had initial values.

- This would leave the array elements uninitialized.

- This is A Bad Thing.

Initializing Array Elements Upon Declaration

Curly braces!

- e.g. float prices[30] = { 10.99, 20.99 };

If you initialize at least one element, all uninitialized ones are set to 0.

Thus, the example would initialize

prices[0] to 10.99, prices[1] to 20.99,

and all other elements to 0.0.

Initializing Array Elements After Declaration

Tediously, ...

one ...

at ...

a ...

time.

Initializers

You can initialize
the entire array
or
just part of it if you want.

You can even leave out the size of the array if you have an initializer list:

- e.g. float prices[ ] = {10.99, 20.99 };

If you have more initializers than you said you would have in the size, you will get a compiler warning or error.

- e.g. float prices[2] = { 10.99, 20.99, 30.99 };

"Invalid Aggregate Initialization"

You can only do this type of initialization of the entire array when you declare the array variable.

Displaying Arrays

Need to print the individual elements.

- e.g. printf("%d\n", prices); /* won't work as you want it to */

Typical method:

```
i = 0;
while( i < 30 )
{
    printf(“#%d is %d\n", i, prices[i]);
    ++i;
}
```

Summary

- Arrays are useful for keeping track of groups of similar data
- You use square brackets for the size and index

- Initialize your arrays (or at least one element) upon declaration