3D Arrays - Example

• To define a 3D array which holds prices for ice-creams:

```cpp
const int Flavors = 12,
    Sizes = 3,
    Conetypes = 2;
float Prices[Flavors][Sizes][Conetypes];
```

3D Arrays - Example

• To address individual elements

```cpp
cout << Prices[8][2][1];
// price of flavor 9, size 3, conetype 2.
Price[10][0][1] = 4;
// price of flavor 11, size 1, conetype 2.
```

3D Arrays - Example (cont …)

• If you wish to loop through higher dimensional arrays displaying or setting all elements, then you require a loops for each dimension.
• For 2D arrays, you need 2 nested loops.

3D Arrays - Example (cont …)

• For 3D arrays, you need 3 nested loops:

```cpp```
```cpp
for(int f = 0; f < Flavors; f++)
    for(int s = 0; s < Sizes; s++)
        for(int c = 0; c < Conetypes; c++)
            Price[f][s][c] = 5;
// loop thru 12*3*2 (72) times
```

3D Arrays - Example (cont …)

• It is easy to visualize the array in three dimensions:

```
Size

Flavor

Conetype
```

Greater than 3D arrays

• You may declare and use greater than 3D arrays, but these are hard to visualise at first.
• In business, you may well be dealing with 7 dimension, 10 dimension, or higher arrays.
• Just think as each of the elements of the array as having more than three attributes which describes it uniquely.