

C-style arrays & vectors

C-Style Arrays

If we want to keep multiple values in one variable we need to use an array. An array is an object that allows us to keep numerous items of the same type together in one object. For example if we want to store 50 integer values:

```
int lots[50];
```

This creates an array called *lots* that contains 50 integer values. Each individual value in the array is access by a subscript, just like we did with apstrings. The first location in the array is called the 0th location, the second location in the array is called the 1st location, etc.

```
cout << lots[0];    // Prints out first location in the array
cout << lots[5];    // Prints out 6th location in the array
```

The following code prints out every element of the array.

```
for (int i = 0; i < 50; i++)
    cout << lots[i];
```

What happens if we access location 75?

```
x = lots[75];
```

This will cause the program to access a location that does not exists. Unpredictable things could happen. The computer could crash immediately, or could crash later executing something that does not appear connected to the above statement. C-style arrays are called "Unsafe Arrays" because we can access any location with a subscript, even though that location may not exist in the array.

Another thing we cannot do with C-style arrays is changed the size of the array. Once we give it a size, it cannot be increased or decreased. We are stuck with that size.

Vectors

Vectors are similar to arrays in that they store multiple values of the same type. But with vectors we have the advantage of subscript checking plus we can also change the size of the vector. We also have some vector member functions to manipulate the vector.

To use a vector we first must include the class:

```
#include "apvector.h"
```

To declare a vector of 50 integers

```
apvector<int> lots(50);
```

We access the vector just like we would access an array.

```
cout << lots[0];    // Prints out first location in the vector
cout << lots[5];    // Prints out 6th location in the vector
```

What happens if we access location 75?

```
x = lots[75];
```

Now the program will stop execution and give an index out of range error. The index 75, is not in the valid range of indices 0 - 49.

We also can change the size of the vector.

```
lots.resize(60);
```

Now the vector size is 60 elements long.

If you do not know how big the vector is you can use the *length* function to determine it's size.

```
cout << "The vector is " << lots.length() << " elements" << endl;
```

The following code prints out every element of the vector.

```
for (int i = 0; i < lots.length(); i++)  
    cout << lots[i];
```

You can also initialize all vector locations to one value. If you want to declare a 10 element vector with all vector locations starting with the value of 28, do as follows:

```
apvector<int> a(10, 28);
```