CS 162C++
Accessing Arrays using Pointers

When we talked about arrays, we said that the name of an array is actually the address of the start of the array. For this reason, pointers can be used to access arrays very simply.

Array Addresses and Pointers

First, let's create a simple array in main and display it.

```
#include <iostream>

using namespace std;

int main()
{
   const int SIZE = 10;
   int theArray[SIZE] = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19};

   for (int i = 0; i < 6; i++)
      cout << theArray[i] << " ";

   cout << endl;

   return 0;
}
```

Now, add a pointer to the program and set it equal to the address of the first element in the array. There are two ways to do this, as shown below.

```
#include <iostream>

using namespace std;

int main()
{
   const int SIZE = 10;
   int theArray[SIZE] = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19};

   // set pointer equal to start of array using name
   int * ptr1 = theArray;
   // set pointer equal to start of array using address of first element
   int * ptr2 = &(theArray[0]);

   cout << ptr1 << endl;
   cout << ptr2 << endl;

   cout << endl;

   return 0;
}
```

Output:

```
0x7ffeefbff500
0x7ffeefbff500

Program ended with exit code: 0
```

Accessing an Array with a Pointer

The following program shows how you can use a pointer to display the elements of an array. Note that when we increment the pointer, it is being incremented by the size of the data element. If we are on a computer where the size of an integer is 4 bytes, then the pointer is incremented by 4 bytes each time. If the size of an integer is 8 bytes, then the pointer is incremented by 8 bytes each time.

```
#include <iostream>

using namespace std;

int main()
{
   const int SIZE = 10;
   int theArray[SIZE] = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19};

   // set pointer equal to start of array using name
   int * ptr = theArray;

   for (int i = 0; i < SIZE; i++)
   {
      cout << *ptr << " ";
      ptr++;
   }

   cout << endl;

   return 0;
}
```

We could have combined this into a single for loop as follows:

```
#include <iostream>

using namespace std;

int main()
{
   const int SIZE = 10;
   int theArray[SIZE] = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19};

   for (int * ptr = theArray; ptr < theArray+SIZE; ptr++)
      cout << *ptr << " ";

   cout << endl;

   return 0;
}
```

Two other ways of coding this same loop are:

```
#include <iostream>

using namespace std;

int main()
{
   const int SIZE = 10;
   int theArray[SIZE] = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19};

   // set pointer equal to start of array using name
   int * ptr = theArray;

   // output using dereference operator
   for (int i = 0; i < SIZE; i++)
      cout << *(ptr + i) << " ";
   cout << endl;

   // output using subscripting operator
   for (int i = 0; i < SIZE; i++)
      cout << ptr[i] << " ";
   cout << endl;

   return 0;
}
```

Not that using the dereference operator * is equivalent to using the subscript operator []. It is purely a matter of personal choice which way to go.