CS 162C++
Functions with Arrays and Pointers

In a previous module, we looked at how to pass arrays to functions. You simply define the parameter as being an array parameter and pass the array. Now let's consider how to pass arrays that have been dynamically created using pointers.

Passing arrays to functions with pointers

If you create an array in main statically, you can pass it to a function in two ways. One is using an array parameter as we covered before, the other is by using a pointer.

```
#include <iostream>
#include <iomanip>    // For output formatting
using namespace std;

void display(int theArray[], int size)
{
   const int LINE = 5;

   for(int i = 0; i < size; i++)
   {
      cout << setw(5) << theArray[i];

      if ((i+1) % LINE == 0)
         cout << endl;
   }

   cout << endl;
}

int main()
{
   const int SIZE = 10;

   int values[SIZE] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29};

   display(values, SIZE);

   return 0;
}
```

Notice that the display function will work with **any** integer array, since the size is passed in as a parameter and not defined as part of the function. Also, note that when we call display, we simply provide the name of the array to it. Now, let's pass the array using a pointer.

```
#include <iostream>
#include <iomanip>    // For output formatting
using namespace std;

void display(int *theArray, int size)
{
   const int LINE = 5;

   for (int i = 0; i < size; i++)
   {
      cout << setw(5) << *(theArray+i);

      if ((i+1) % LINE == 0)
         cout << endl;
   }
   cout << endl;
}

int main()
{
   const int SIZE = 10;

   int values[SIZE] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29};
   int * ptr = values;

   display(ptr, SIZE);

   return 0;
}
```

The only real difference is that we defined the parameter as an

```
int * theArray
```

instead of

```
int theArray[]
```

I chose to do the displaying of the function using a dereference

```
*(theArray +i)
```

instead of

```
theArray[i]
```

But that is just a syntax choice, either would work fine.

Returning an array from a function

In order to successfully create an array in a function and return it to main, you need to create it on the heap using new. As a test, you can run this sample code and see how it does not work properly. This is because memory allocated to a local-variable in a function goes away at the end of the function.

Bad Version

```
#include <iostream>

using namespace std;

int * create()
{
   int theArray[] = {1, 3, 5, 7, 9, 11};
   return theArray;
}
int main()
{
   int *ptr = create();

   for (int i = 0; i < 6; i++)
      cout << *(ptr +i) << endl;

   return 0;
}
```

On my computer, running this program returns:

```
1
32766
3479
1
3488
1
Program ended with exit code: 0
```

Instead, you want to do it this way:

Good Version

```
#include <iostream>

using namespace std;

int * create()
{
   int *theArray = new int[6];
   for (int i = 0; i < 6; i++)
      *(theArray + i) = 2 * i + 1;

   return theArray;
}
int main()
{
   int *ptr = create();

   for (int i = 0; i < 6; i++)
      cout << *(ptr +i) << endl;

   delete [] ptr;

   return 0;
}
```

Note that you have to release the memory using delete at the end of main, after you are done with it.