CS 162C++
Passing Arrays to Functions

Now that you are comfortable working with one and two-dimensional arrays, this document will present how you can pass them to functions.

Passing 1D arrays to functions

First, let's consider 1D arrays since they are simpler. The following is an example of defining an array and printing it out in main:

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

int main()
{
   const int SIZE = 10;
   const int LINE = 5;

   int values[SIZE] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29};
      for(int i = 0; i < SIZE; i++)
      {
         cout << setw(5) << values[i];

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

   cout << endl;
   return 0;
}
```

Now create a function display and move the output code into it and you get this **incorrect version**:

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

void display()
{
   const int LINE = 5;

   for(int i = 0; i < SIZE; i++)
   {
      cout << setw(5) << values[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();
   return 0;
}
```

If we tried to compile and run this, the compiler would complain of undefined variables values and SIZE in the function display. Since they are defined in main, we need to pass them in as parameters. We also need to tell the compiler that values is an array, not just an integer variable. With these changes, the final code looks like:

```
#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.

Two-dimensional Arrays

Let's do the same sort of process to create a two-dimensional array and display it. Consider the following program that calculates and displays a multiplication table from 0x0 to 9x9.

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

int main()
{
    const int ROWS = 10;
    const int COLS = 10;
    int mults[ROWS][COLS];

    for(int i = 0; i < ROWS; i++)
        for(int j = 0; j < COLS; j++)
            mults[i][j] = i * j;

    for(int i = 0; i < ROWS; i++)
    {
        for(int j = 0; j < COLS; j++)
            cout << setw(5) << mults[i][j];
        cout << endl;
    }

    cout << endl;
    return 0;
}
```

If we do the same sort of moving it to a function, you would get this **incorrect version**:

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

void display(int theArray[][], int rows, int cols)
{
    for(int i = 0; i < rows; i++)
    {
        for(int j = 0; j < cols; j++)
            cout << setw(5) << theArray[i][j];
        cout << endl;
    }

    cout << endl;
}
int main()
{
    const int ROWS = 10;
    const int COLS = 10;
    int mults[ROWS][COLS];

    for(int i = 0; i < ROWS; i++)
        for(int j = 0; j < COLS; j++)
            mults[i][j] = i * j;

    display(mults, ROWS, COLS);

    return 0;
}
```

The compiler complains about this. The reason is that the array is two dimensional in your program, but in memory it is one-dimensional. The compiler needs to know the length of each row in order to figure the proper address. We will look at this in more detail next week in the section on flattening an array. But what you have to know now, is that in the function declaration, you have to specify the length of each row – which is COL. So, the final, working, program is:

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

const int COLS = 10;

void display(int theArray[][COLS], int rows, int cols)
{
    for(int i = 0; i < rows; i++)
    {
        for(int j = 0; j < cols; j++)
            cout << setw(5) << theArray[i][j];
        cout << endl;
    }

    cout << endl;
}
int main()
{
    const int ROWS = 10;
    int mults[ROWS][COLS];

    for(int i = 0; i < ROWS; i++)
        for(int j = 0; j < COLS; j++)
            mults[i][j] = i * j;

    display(mults, ROWS, COLS);

    return 0;
}
```