CS 162C++
Passing parameters to functions by value, reference, and pointer reference

When you are passing something to a function, you can either pass by value or reference; as you learned in CS 161. Passing by value means that a copy is made and the copy is then provided to the function. Pass by reference means that you pass the address of the item to the function.

Now that you are learning about pointers, we are going to look at how you can use pointers as another way to do a pass by reference.

Pass by Value

The simplest approach is simply pass by value. The problem with this is that it requires copying whatever we are passing and thus is less efficient as we will discuss when we start working with objects.

```
//
//  main.cpp
//  passing by reference
//

#include <iostream>
#include <iomanip>  // for output formatting

using namespace std;

void display(int x)
{
    cout << "the value is " << x << endl;
}

int main()
{
    int a = 6;

    display(a);

    return 0;
}
```

In this example, we create an integer and pass it to the display function. The result is that the value is displayed, just as we would expect.

Pass by Reference

When we do pass by reference, we use an ampersand to indicate in the function declaration that the compiler is to set up an alias to the variable being passed instead of copying it. That is, in this program both x and a are names for the same memory location. In this case, there is no benefit to pass by reference and the program proceeds exactly as the one above does.

```
//
//  main.cpp
//  passing by reference
//

#include <iostream>
#include <iomanip>  // for output formatting

using namespace std;

void display(int &x)
{
    cout << "the value is " << x << endl;
}

int main()
{
    int a = 6;

    display(a);

    return 0;
}
```

To see the benefit of pass by reference, let's look at an example where it is necessary to modify the original memory location in order for the program to work properly. If you tried to code exchange without pass by reference, it would not work.

```
//
//  main.cpp
//  passing by reference
//

#include <iostream>
#include <iomanip>  // for output formatting

using namespace std;

void exchange(int &x, int &y)
{
    int temp = x;
    x = y;
    y = temp;
}

int main()
{
    int a = 6, b = 7;

    cout << "a is " << a << " and b is " << b << endl;
    exchange(a, b);
    cout << "a is " << a << " and b is " << b << endl;

    return 0;
}
```

Pass by Pointer

The final way to pass something to a function is to pass it as a pointer. This is similar to passing by reference, but instead of using a reference or alias to the item, you use a pointer, which is a variable that holds an address. Here is the same program as above, but with pointers.

Note that the function parameters are defined as pointers, when you call the function, you need to pass in addresses, and the function itself must use dereferencing to get to the contents of the variables.

```
//
//  main.cpp
//  passing by pointer reference
//

#include <iostream>
#include <iomanip>  // for output formatting

using namespace std;


void exchange(int *x, int *y)
{
    int temp = *x;
    *x = *y;
    *y = temp;
}

int main()
{
    int a = 6, b = 7;

    cout << "a is " << a << " and b is " << b << endl;

    exchange(&a, &b);

    cout << "a is " << a << " and b is " << b << endl;

    return 0;
}
```