Function Parameter and Pointers

In C++, you can use pointers as function parameters to pass the memory address of a variable to a function. This can be useful when you want to modify the value of a variable in a function without returning it.

Here is an example of how you can use pointers as function parameters in C++:

```
#include <iostream>

void increment(int* x)
{
    (*x)++;  // increment the value stored at the memory addr
}

int main()
{
    int x = 5;
    std::cout << "x before increment: " << x << std::endl;

    increment(&x);  // pass the memory address of x to the in

    std::cout << "x after increment: " << x << std::endl;

    return 0;
}
```

Output

```
x before increment: 5
x after increment: 6
```

the function increment takes a pointer to an int as a parameter, and modifies the value stored at the memory address pointed to by the pointer. This has the effect of modifying the value of x in the main function, even though x is not returned by the increment function.

Using pointers as function parameters can be a powerful tool in C++, but it's important to be careful when working with pointers, as they can be tricky to work with and can lead to problems if used incorrectly. It's always a good idea to check that pointers are valid and initialized before dereferencing them, and to deallocate memory when you're done with it to avoid memory leaks.