# References

void swap(int a, int b) {

// Incoming parameters are copied

int c = a;

a = b;

b = /\*a\*/ c;

}

void main() {

int a=1;

int b=2;

swap(a,b); // Does not change my locals

cout << a << " " << b << endl;

system("pause");

}

How to fix this?

void swap2(int\* a, int\* b) {

int c = \*a;

\*a = \*b;

\*b = /\* \*a \*/ c;

}

swap2(a,b); // What do we do here?

swap2(&a,&b); // Very obvious that my locals are in danger

You rarely use the pointer without dereferencing it. You almost always have the “\*” with it. If you do a lot of pointer work you quickly tire of the syntax.

When would you not? Keep these in mind for later.

* Pointer math (p = p + 1)
* Array access (p[23]) which is really (\*p+23) pointer math
* Initializing (p = &a)

New to C++ is a way to tell the compiler that a variable should always be treated as an address.

void swap3(int& a, int& b) {

// The compiler knows a and b are addresses and treats them

// differently from "c". You have to keep it in mind without

// the visual clue.

int c = a;

a = b;

b = /\* a \*/ c;

}

swap3(a,b); // Compiler does the “address of”. Not obvious at all!

But you shouldn’t call a function without knowing the signature anyway. You can look at the header file and tell if it address-of or copy.

This is for your convenience (or confusion) the disassembly is identical.

int a=4;

int\* p; // What value is here?

int& q = a; //You must initialize when you create

q[0] = 4; // Can't use index (arrays are passed by REFERENCE anyway)

q=q+1; // Can't use pointer math

int doCube(int a) {

a = a\*a\*a;

cout << a << endl;

return a;

}

void main() {

int a=4;

int b;

b = doCube(a);

cout << "Cube of " << a << " is " << b << endl;

system("pause");

}

Add the “&” to the function and rerun. The function is changing the incoming parameter, which isn’t a copy.

int doCube(const int& a) // "const"

The “const” means you promise not to change “a”. The compiler won’t let you. You, the caller, feel safer. But you aren’t:

int doCube(const int& a) {

int& b = (int &)a;

b = b\*b\*b;

cout << b << endl;

return b;

}

The “const” is just a helper at compile time. It does nothing at the runtime and does not contribute to the assembly. But USE IT. Let the compiler find what it can!