```
void f(int x) { x+=10; }
void g(int& y) { y += 10; }
int main() {
    int x = 10;
    int y = 10;
    f(x);
    g(y);
    cout << x << " " << y;
```

Terminology: Actual parameter. This is the variable or expression that is used as input when a function is called.

Ex:
```
int z = 99;
f(z);
```
actual parameter for this call to f.

Formal parameter. This is the place holder variable used in the function definition.

```
int f(int x) {
```
formal parameter

```
    cout << x << "\n";
    x *= 100;
    :
    ;
}
```

Actual param ≡ when calling ...

Formal param ≡ when writing ...

Question: what is the relationship of the formal param and actual param?

Formal is copy of actual?

Yes for by value.

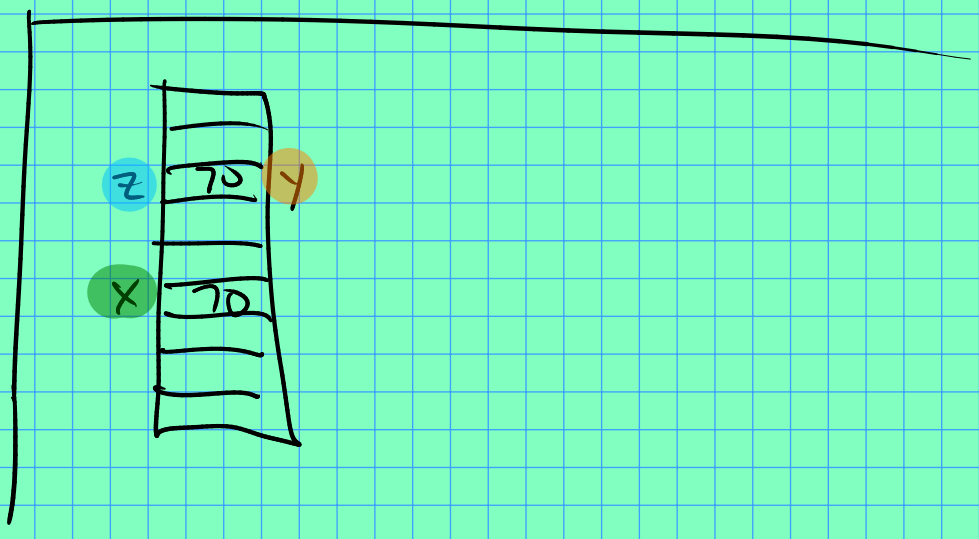
Formal is a synonym of actual?

Yes for by reference.

In pictures:

```
void f(int x) { x+=10; ... }
void g(int& y) { y+=10; ... }
```

```
int main() {
    int z = 70;
    f(z);
    g(z);
    :
    ;
```



New Topic: Vectors

Stores an indexed list of variables (with homogeneous type).

Example: read list of integers from stdin and print them in reverse order.

E.g. echo "1 2 3 4 5" | ./a.out

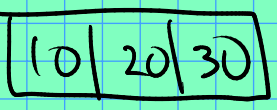
5
4
3
2
1

Impossible using only a fixed # of variables.

Vectors let you store the whole list:

```
while (cin >> n)
    V.push_back(n);
```

say input was 10 20 30

V: 

V.push_back(100);

⇒ V: