More about functions in C/C++.

Say we have the following function:

```
void inc(int x)
{
    x++;
}
```

```
void inc2(int& x)
{   x++;
}
```

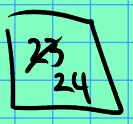
```
int main()
{
    int y;
    y = 23;
    inc(y);
    cout << y << "\n"; // ??
    // prints 23.
    inc2(y); // now y == 24.
```

What is the relationship in memory between x & y?

option 1: (default)


x is a "by value" parameter.

option 2:




x is a "by reference" parameter.

Nomenclature: in the above, x is the "formal parameter".
in the call (applying the function) inc(y),
y is the "actual parameter".

So, with by value, formal param is a copy of the actual param.

With by reference, formal param is a synonym for the actual param.

A few more basic facts about functions:

Multivariate functions:

$f: \mathbb{Z} \times \mathbb{R} \to \mathbb{Z}$ ≈ `int f(int x, double y);`

(math class) ( C/C++)

Question: what about $f: \mathbb{R} \to \mathbb{R} \times \mathbb{R}$ ?

`?[R×R]? f(double t)`

One idea: use by-reference params for output!

`void f(double t, double& y1, double& y2);`

input output.

Overloading functions is also possible:

it's OK to have more than 1 function w/ the same name, as long as you can tell them apart from a function call.

Examples: say we already have

`int f(int x);`

Which of the following could co-exist?

`int f(int x, int y);` ✓

`int f(string x);` ✓

`double f(int x);` ✗