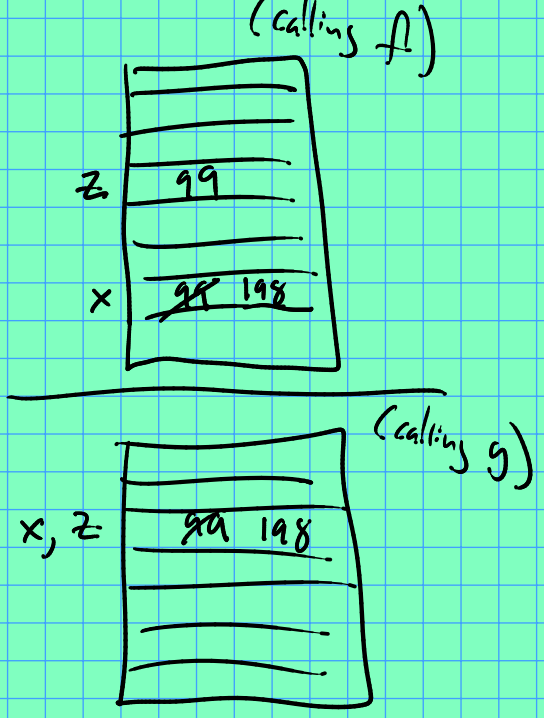
By value vs. by reference

```
int f(int x) {
    x *= 2;
    :
}
int g(int& x) {
    x *= 2;
    :
}
int main() {
    int z = 99;
    f(z);
    g(z);
}
```



Why use value over reference or vice versa?

By reference use cases:

* use parameters for output (say if conceptually your function should return several values)
* passing large datatypes (e.g. a string or vector). By reference avoids copying large amounts of data.

By value use cases:

* everything else.
Avoids certain side effects.

Terminology: actual parameter:
the variable passed into
the function from
"the outside".
E.g. z above in main().
formal parameter: the variable
in the parameter list in
the function.
E.g. x above in f, g.

By value: formal param is a copy of actual

By reference: formal param is a synonym of actual

Common trick: use const by ref. to
efficiently emulate a by value call.

New topic: Vectors.

Motivating example: read integers from stdin
& print them in reverse order:

```
echo {1..10} | ./program
10
9
8
:
1
```

Issue: requires a non-constant # of variables — can't print anything until we've stored everything.

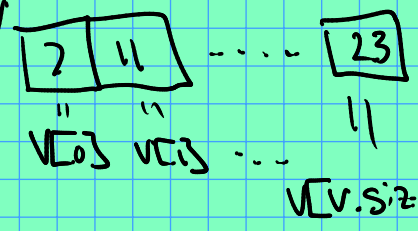
vector is a sequence of variables of homogeneous type.

it can expand to store as many variables as you need, & you need not know a bound at compile time.

```
vector<int> v;
```

~~vector v;~~ // won't compile.

```
v.push_back(7);
v.push_back(11);
   ⋮
v.push_back(23);
```




```
for(i=0; i<v.size(); i++){
   ⋮
}
```

```
int n; // for input
vector<int> v;
while (cin >> n) {
    v.push_back(n);
}
for(int i = v.size()-1; i>=0; i--)
    cout << v[i] << "\n";
```