Separating I/O from Computation

EECS 211

Winter 2018

Good software design

- Correct
- Efficient
- Simple

Code isn't just for computers

In practice, other people need to read it:

- Your boss

Code isn’t just for computers

In practice, other people need to read it:

- Your boss
- Your colleagues

Code isn’t just for computers

In practice, other people need to read it:

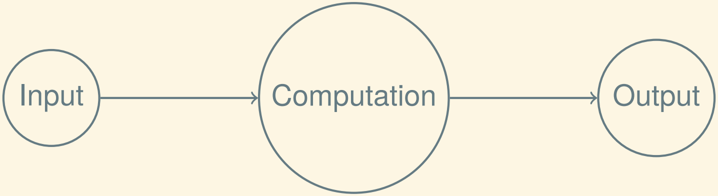
- Your boss
- Your colleagues
- Your successors

Code isn’t just for computers

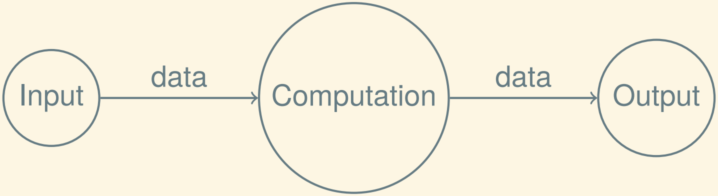
In practice, other people need to read it:

- Your boss
- Your colleagues
- Your successors
- You in the future

Separation of concerns



Separation of concerns



Data must be structured

Bits without structure are meaningless

Two most basic data structures:

- struct
- vector

What they are

- a struct creates a new type of compound of box made of smaller boxes
- a vector is a sequence of any number of boxes of the same type

Struct basics: declaration

To declare a new struct type:

```
struct Posn
{
    double x;
    double y;
};
```

Struct basics: declaration

To declare a new struct type:

```
struct Posn
{
    double x;
    double y;
};

struct Account
{
    long id;
    std::string owner;
    long balance;
};
```

Struct basics: construction

To declare and initialize a struct variable, list the values of the member variables:

```
Posn p{3, 4};
```

Struct basics: construction

To declare and initialize a struct variable, list the values of the member variables:

```
Posn p{3, 4};
```

You can also create a struct without declaring a variable:

```
Posn get_posn()
{
    double x = get_x_coordinate();
    double y = get_y_coordinate();
    return Posn{x, y};
}
```

Struct basics: using

A member variable of a struct is accessed by following the struct with a period and the name of the member variable:

```
Posn p = get_posn();
std::cout << '(' << p.x << ", " << p.y << ')';
```

Struct basics: using

A member variable of a struct is accessed by following the struct with a period and the name of the member variable:

```
Posn p = get_posn();
std::cout << '(' << p.x << ", " << p.y << ')';
```

If you don’t initialize a struct, its fields are uninitialized:

```
Posn p;
z = p.x + p.y;      // Error!
```

Struct basics: using

A member variable of a struct is accessed by following the struct with a period and the name of the member variable:

```
Posn p = get_posn();
std::cout << '(' << p.x << ", " << p.y << ')';
```

If you don't initialize a struct, its fields are uninitialized:

```
Posn p;
z = p.x + p.y;      // Error!
```

However, you can assign them:

```
p.x = 3;
p.y = 4;
```

Vector basics: creating

You can declare a vector with elements similar to how you declare a struct:

```
#include <vector>

std::vector<int> v{2, 3, 4, 5};
```

Vector basics: creating

You can declare a vector with elements similar to how you declare a struct:

```
#include <vector>

std::vector<int> v{2, 3, 4, 5};
```

However, it's more common to build using push_back:

```
std::vector<int> v;
v.push_back(2);
v.push_back(1);
v.push_back(3);
```

v now contains 2, 1, 3.

Vector basics: size

The size *member function* returns the number of elements:

```
for (size_t i = 0; i < v.size(); ++i)
      std::cout << v[i] << '\n';
```

Vector basics: size

The size *member function* returns the number of elements:

```
for (size_t i = 0; i < v.size(); ++i)
      std::cout << v[i] << '\n';
```

Note! The number of elements is one more than the last index.

Vector basics: empty

The empty member function returns whether a vector is empty:

```
if (grades.empty())
    std::cout << "No grades were entered.";
```

Vector basics: access

Reverse a vector:

```
for (size_t i = 0; i < v.size() / 2; ++i) {
      size_t j = v.size() - i - 1;
      int temp = v[i];
      v[i]        = v[j];
      v[j]        = temp;
}
```

Vector basics: iteration

Can you spot the bug?

```
double sum = 0.0;

for (size_t i = 0; i <= v.size(); ++i)
    sum += v[i];
```

Vector basics: iteration

Can’t overrun the bounds when using for-each syntax:

```
double sum = 0.0;

for (double vi : v)
    sum += vi;
```

To the terminal!