C++ at Velocity, Part X

CS 002 - WI 2015

The `struct`

- Allows us to define compound data types

```
struct point {
  double x;
  double y;
}; // MUST have semicolon

point p; // in C, use 'struct point'
p.x = 3.0; // 'dot notation'
p.y = 4.0;
printf("dist %f\n",
          sqrt(p.x * p.x + p.y * p.y));
```

(C 6.15-17)

The `struct`

- Can include pointers to other structs

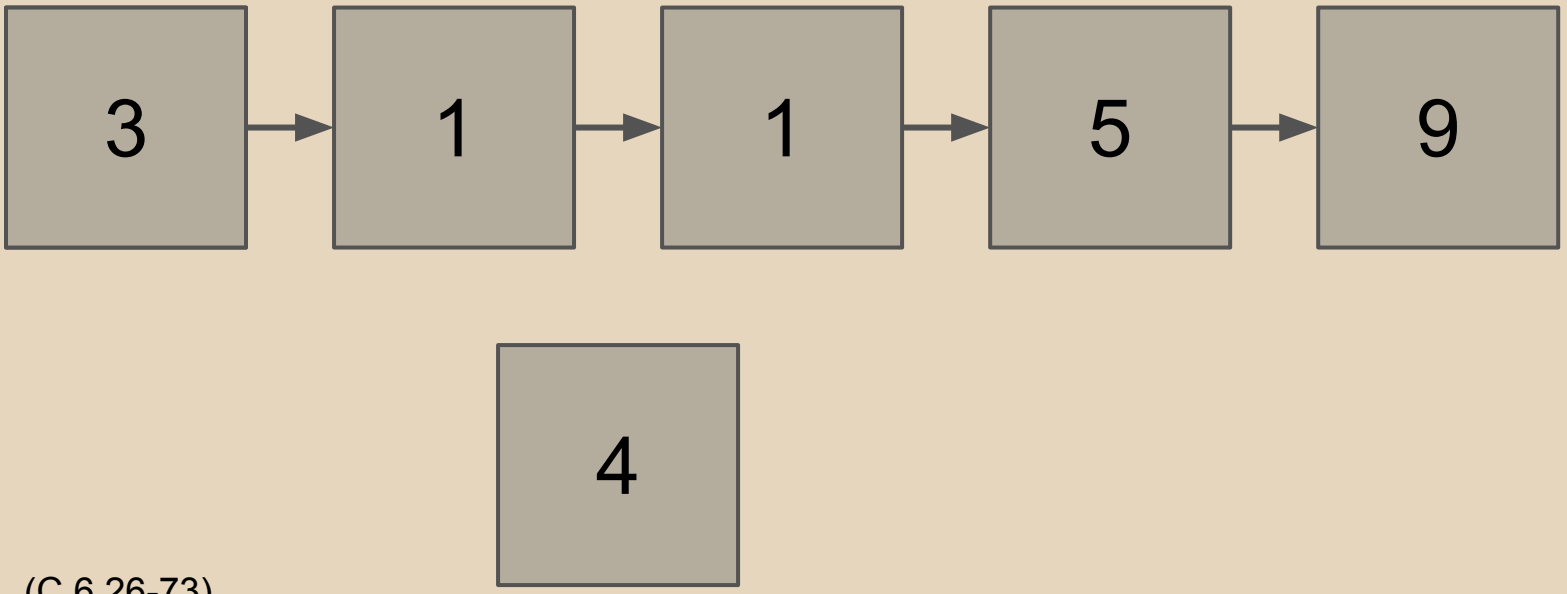
```
struct node {
    int value;
    node * next;
}

void append_to(node * self, node * prev)
{
    prev->next = self; // 'arrow' notation
}
```

(C 6.19-21)

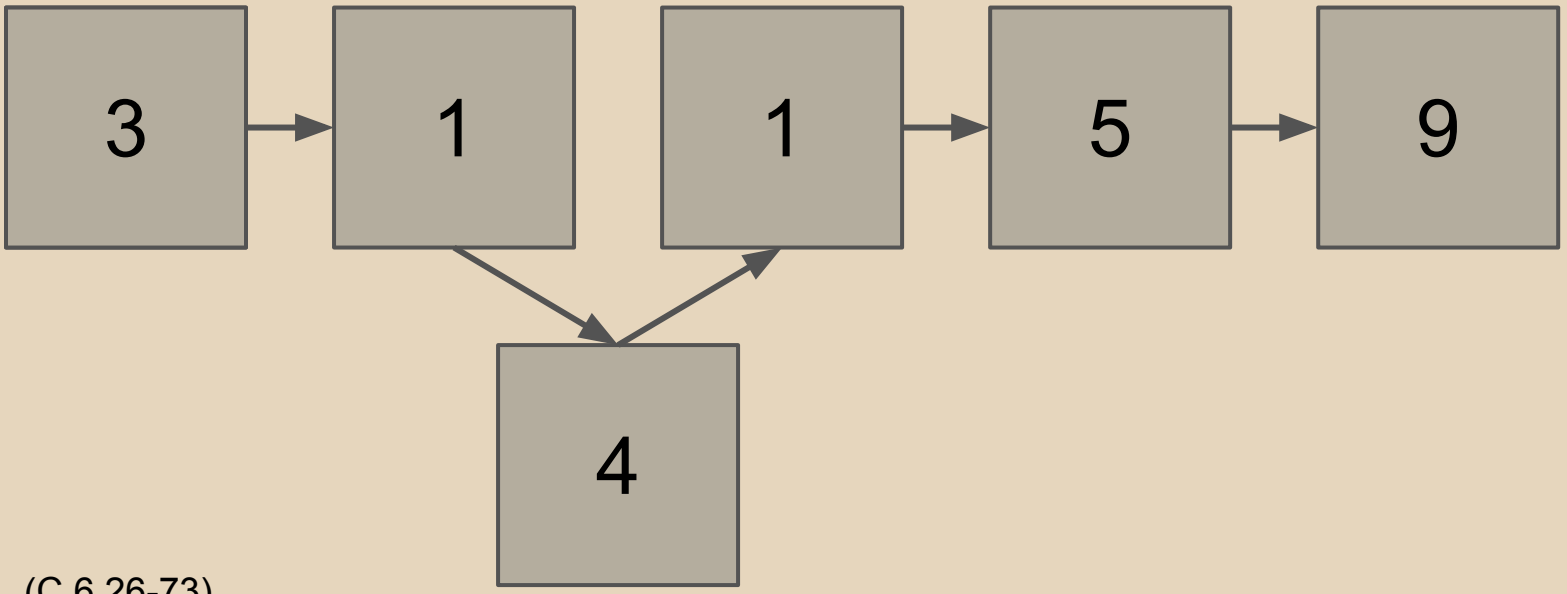
Linked lists

- Simple 'complex' data structure
- Supports iteration, constant-time insertion, constant-time deletion
- No constant-time random access!



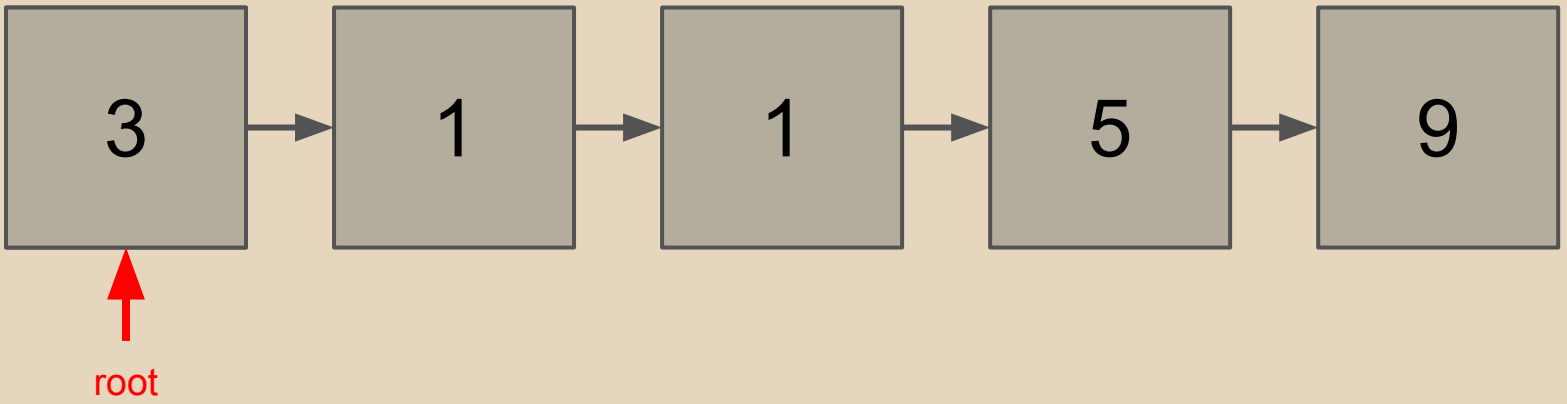
Linked lists

- Simple 'complex' data structure
- Supports iteration, constant-time insertion, constant-time deletion
- No constant-time random access!



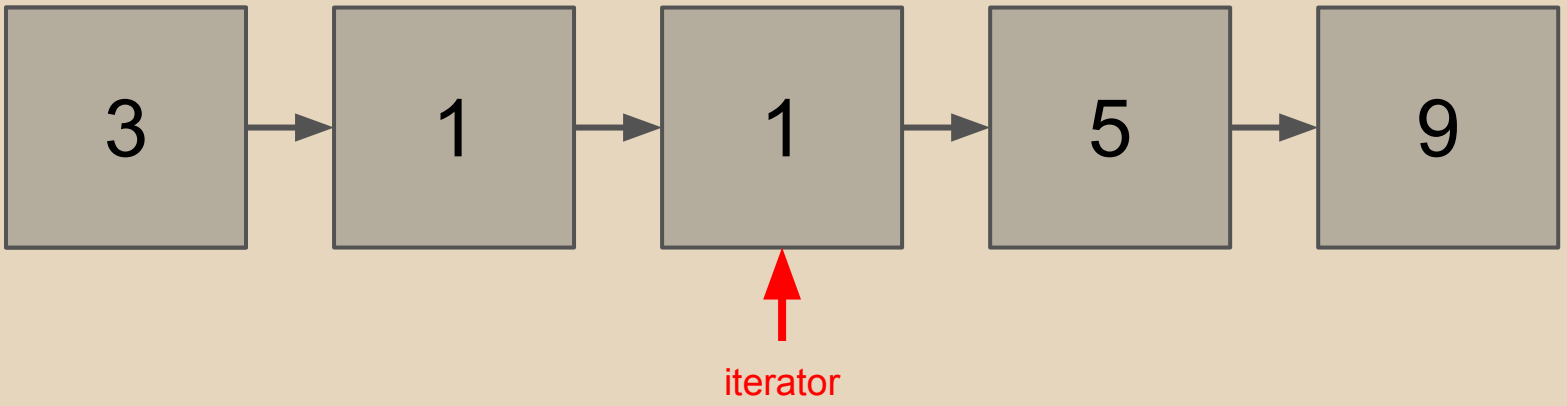
Linked lists

- Simple 'complex' data structure
- Supports iteration, constant-time insertion, constant-time deletion
- No constant-time random access!



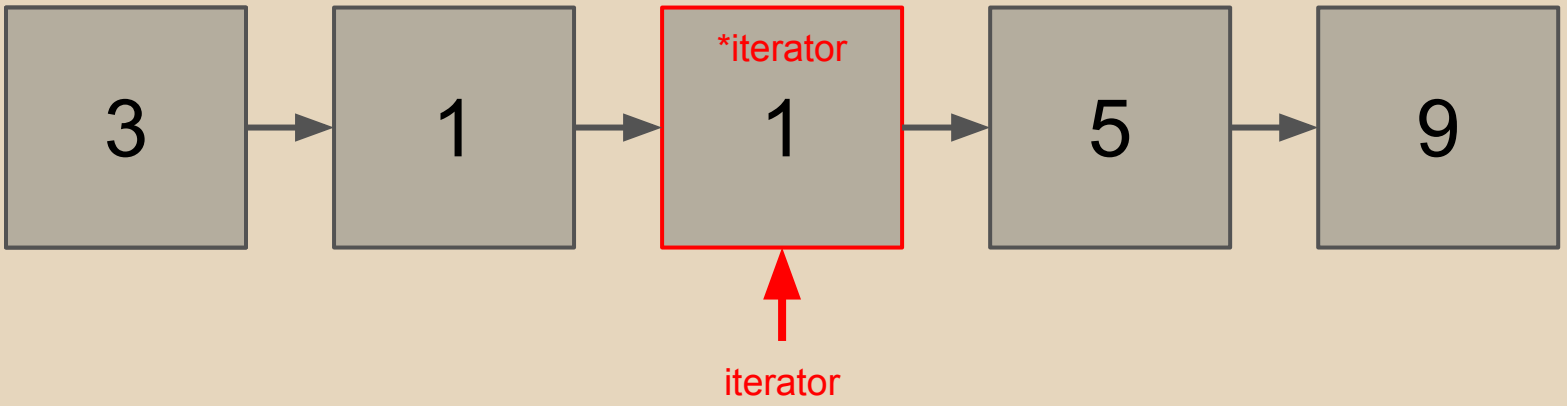
Linked lists

- Simple 'complex' data structure
- Supports iteration, constant-time insertion, constant-time deletion
- No constant-time random access!



Linked lists

- Simple 'complex' data structure
- Supports iteration, constant-time insertion, constant-time deletion
- No constant-time random access!



Linked lists

- With some creativity we can come up with ways to...
  - Add elements
  - Delete elements
  - Retrieve elements by index
  - Print elements in order
- But we're still working with 'loose' data structures
- Want some way to neatly bundle functionality with data

Templates

- Recall that C++ is a **statically typed** language.
- Recall our linked-list example from earlier.
  - Linked list of integers
- What if we want a linked list of `Vector2`?
- Don't want to copy a whole bunch of code!
  - Where possible, don't repeat yourself
  - What if we copy an implementation with bugs?

Templates

- Templates allow us to apply one code pattern to many data types.
- Templates take one or more types as "parameters".

```
template <class T>
T sum(T a, T b)
{
    T result = a + b;
    return result;
}
...
printf("%d\n", sum<int>(8, 11));
```

(C++ 6.20-31)

Templates

- Classes and structs can also be templated.

```
template <class T>
struct node
{
   T data;
   node<T> * next;
};
...
node<double> n;
n.data = 3.14159;
```

Templates

- N.B.: As a general rule, template *classes* must be completely defined in header file!

```
template <class T>
class Vector2
{
...
public:
    Vector2(T x, T y)
    {
        this->x = x;
        this->y = y;
    }
...
}
```