CS 334
Lecture 17
C++

* 1979
* Bjarne Stroustrup
* C + objects.

Themes
- justification is crucial.
- compromise when necessary.

Goal
- “Better than C” +
  Data Abstraction +
  OOP
Rules

- Useful now.
- Support lots of style.
- Orthogonality of Design
- No implicit static type errors.
- Support user defined types as well as built-in types.
- Zero overhead Rule
  - Don't pay for feature you don't use.
C++ Features

Point Code

Point *p = new Point (1, 2);
p = move (10, 10) \rightarrow \text{virtual}
x = p \Rightarrow \text{get X()}; \rightarrow \text{non virtual.}

Virtual: dynamic lookup based on runtime type of receiver.
Non Virtual: resolved statically based on declared type of receiver.
+ performance.
+ prevent subclasses from changing behavior.
PointRep

\[ p \Rightarrow \text{move}(10, 10) ; \]
\[ \Rightarrow \quad \text{fn} = p \Rightarrow \text{utable}[0] \]
\[ \quad \text{call fn}(p, 10, 10) \]

\[ p \Rightarrow \text{getX}(); \]
\[ \Rightarrow \quad \text{call Paint::getX}(p) \]
**Color Point (code)**

```cpp
ColorPoint *cpt = new CPoint(1, 3, RED);
```

1) `cpt -> move (10, 10)`
   
   ```cpp
eq \hspace{2cm} \Rightarrow \hspace{2cm} f_n = \text{cpt} \Rightarrow \text{Vtable}[0]; \hspace{2cm} \text{call} \hspace{2cm} f_n(\text{cpt}, 10, 10) \}; \text{Same!}
   ```

2) `cpt -> get X()`

   ```cpp
   \Rightarrow \hspace{2cm} \text{call Point::getX(cpt)}; \text{Same}
   ```

3) `cpt -> darken()`

   ```cpp
   \Rightarrow \hspace{2cm} f_n = \text{cpt} \Rightarrow \text{Vtable}[1]; \hspace{2cm} \text{call} \hspace{2cm} f_n(\text{cpt});
   ```

- Fields at same offset in obj.
- Methods at same offset in Vtable.
Smalltalk vs. C++ Dyn. Dispatch

Smalltalk:

```
pt <- ... ;
pt moveDx: 5 Dy: 5
```

- `pt` can be anything w/ moveDx: Dy: in its protocol.
- No info about pts dictionary layout.

```
A moveDx: Dy   B print
\   /
\  /  
\ /   
C moveDx: Dy print
```
C++:

```cpp
Pt n Pt = ... ;
Pt -> move (10, 10);
```

will be a `Point` or a subclass of `Point`.

always at same index

More static type info!

+ more efficient dispatch.

- expressiveness.