# CS 225

**Data Structures**

*February 2 – C++ Inheritance*

*G Carl Evans*

## Inheritance

# Derived Classes

**[Public Members of the Base Class]:**

**main.cpp**

```
int main() {
  Square sq;
  sq.getLength(); // Returns 1, the length init'd
                  // by Shape's default ctor
  ...
}
```

**[Private Members of the Base Class]:**

## Polymorphism

*The idea that a single interface my take multiple types or that a single symbol may be different types.*

*In Object-Orientated Programming (OOP) a key example is that a single object may take on the type of any of its base types.*

## Virtual

## Method Dispatch

1) Look at the type the method is called on
2) Look for the method in that type if found
   A. If type is virtual use runtime type and goto 2 ignoring virtual from now on
   B. Use method that method
3) No method found change to base type and goto 2

Cube.cpp

```
Cube::print_1() {
  cout << "Cube" << endl;
}

Cube::print_2() {
  cout << "Cube" << endl;
}

virtual Cube::print_3() {
  cout << "Cube" << endl;
}

virtual Cube::print_4() {
  cout << "Cube" << endl;
}
```

Cube.h

```
// In .h file:
virtual print_5() = 0;
```

RubikCube.cpp

```
// No print_1() in RubikCube.cpp



RubikCube::print_2() {
  cout << "Rubik" << endl;
}

// No print_3() in RubikCube.cpp



RubikCube::print_4() {
  cout << "Rubik" << endl;
}

RubikCube::print_5() {
  cout << "Rubik" << endl;
}
```

# Runtime of Virtual Functions

| virtual-main.cpp | `Cube c;` | `RubikCube c;` | `RubikCube rc;` `Cube &c = rc;` |
|---|---|---|---|
| `c.print_1();` | | | |
| `c.print_2();` | | | |
| `c.print_3();` | | | |
| `c.print_4();` | | | |
| `c.print_5();` | | | |

## Abstract Class:

**[Requirement]:**

**[Syntax]:**

**[As a result]:**

**virtual-dtor.cpp**

```cpp
class Cube {
  public:
    ~Cube();
};

class RubikCube : public Cube {
  public:
    ~RubikCube();
};
```

## Templates

## template1.cpp

```cpp


T maximum(T a, T b) {
  T result;
  result = (a > b) ? a : b;
  return result;
}
```

List.h

```cpp
#pragma once



class List {
  public:








  private:



};

#endif

```

List.cpp

```cpp
```