CS 225

#9: Inheritance and Templates

September 13, 2021 · *G Carl Evans*

Virtual

- The **virtual** keyword allows tells the system to expect to override the behavior of a class by its derived type.

Example:

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;
}

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

RubikCube.cpp

```
// No print_1()



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

// No print_3()



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

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

Method Dispatch Rules

1)

2)

3)

	`Cube c;`	`RubikCube c;`	`RubikCube rc;` `Cube &c = rc;`
`c.print_1();`			
`c.print_2();`			
`c.print_3();`			
`c.print_4();`			
`c.print_5();`			

Polymorphism

Object-Orientated Programming (OOP) concept that a single object may take on the type of any of its base types.

- A **RubikCube** may polymorph itself to a Cube
- A Cube can<u>not</u> polymorph to be a **RubikCube** *(base types only)*

Why Polymorphism? Suppose you're managing an animal shelter that adopts cats and dogs:

Option 1 – No Inheritance

animalShelter.cpp

```
1  Cat & AnimalShelter::adopt() { ... }
2  Dog & AnimalShelter::adopt() { ... }
3  ...
```

Option 2 – Inheritance

animalShelter.cpp

```
1  Animal & AnimalShelter::adopt() { ... }
```

Pure Virtual Methods

In `Cube`, `print_5()` is a **pure virtual** method:

Cube.h

```
1  virtual Cube::print_5() = 0;
```

A pure virtual method does not have a definition and makes the class and **abstract class**.

Abstract Data Types (ADT):

List ADT - Purpose	Function Definition

List Implementation
What types of List do we want?

C++ Templates:

1.

2.

3.

Templated Functions:

functionTemplate1.cpp

```
1
2
3  T maximum(T a, T b) {
4    T result;
5    result = (a > b) ? a : b;
6    return result;
7  }
```

Where to put templated code?

Templated Classes:

List.h

```
1  #pragma once
2
3
4  class List {
5    public:
6
7
8
9
10
11
12   private:
13
14
15 };
```

List.hpp

```
1
2
3
4
5
```

Two Basic Implementations of List:

1.

2.

Linked Memory:

List.h

```
28 class ListNode {
29   T & data;
30   ListNode * next;
31   ListNode(T & data) : data(data), next(NULL) { }
32 };
```

CS 225 – Things To Be Doing:

1. mp_stickers extra credit due Today!
2. Daily POTDs