WEEK 12

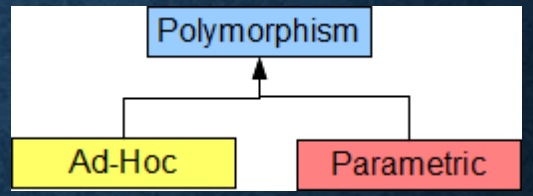
Polymorphism Overview

OVERVIEW

- Week 12-1
  - Polymorphism Categories
    - AD-HOC
      - Coercion
      - Overloading
    - Universal
      - Inclusion
      - Parametric

POLYMORPHISM CATEGORIES

- Christopher Strachey (1967) introduced the concept of polymorphism into procedural programming languages by distinguishing functions in two ways:
- Works differently on different types
  - (AD-HOC)
- Works the same on different types
  - (Parametric)

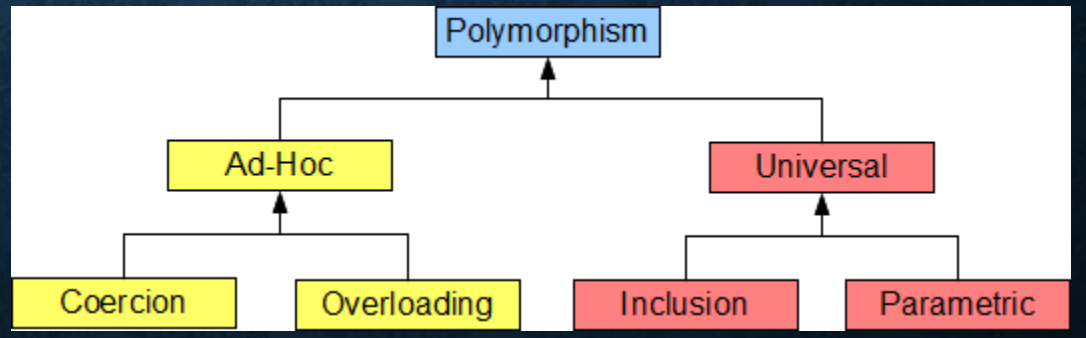


- These two categories were then further broken down and refined into four categories by Cardelli & Wegner (1985)

POLYMORPHISM CATEGORIES (CONT'D)

- The four categories are aligned like so:
- AD-HOC (Works on a finite set of different & potentially unrelated types)
  - Coercion
  - Overloading

- UNIVERSAL (Works on a potentially infinite number of types across some common structure)
  - Inclusion
  - Parametric



AD-HOC - COERCION

- Coercion addresses the differences between argument types in a function call and the parameter types in the functions definition.
- It allows convertible type changes in the argument's type to match the parameter.
- It's an operation (implemented at compile time in C++, compiler inserts conversion code before function call) that allows for the avoidance of a type error.

AD-HOC - COERCION

- Two possible variations of coercion:
  - Narrow the argument type
  - Widen the argument type

COERCION EXAMPLE

```
//  coercion.cpp
#include <iostream>
void display(int a){ std::cout << "Arg is:" << a << std::endl; }
int main() {
        display(10); // Normal use
        display(12.6); // Narrows the argument from double to int – Prints 12
        display('A'); // Widens/Promotes the argument from char to int – Prints 65
}
```

AD-HOC - OVERLOADING

- Overloading addresses the variations of a function's signature.
- This feature allows for binding of function calls with the same identifier but different arguments with the matching definitions (IE function overloading).
- Overloading eliminates multiple function definition errors
- C++ implements overloading at compile type by renaming each identically named function with a distinct identifier.

UNIVERSAL - INCLUSION

- Inclusion allows for multiple member functions of a similar identifier by selecting the definition from a set of definitions based the object's type. Recall the idea of dynamic binding and the virtual keyword.
- The term inclusion refers to the base type including the derived types in a hierarchy and the ability to select the correct function for an object in that hierarchy based on function binding.

UNIVERSAL – PARAMETRIC (GENERIC)

- Parametric addresses the differences between argument types in the function call and the parameter types in the function's definition.
- This allows for function definitions that share identical logic independent of type.
- Recall that this is implemented in C++ at compile time through the use of templates.