19CSE201 :Advanced Programming

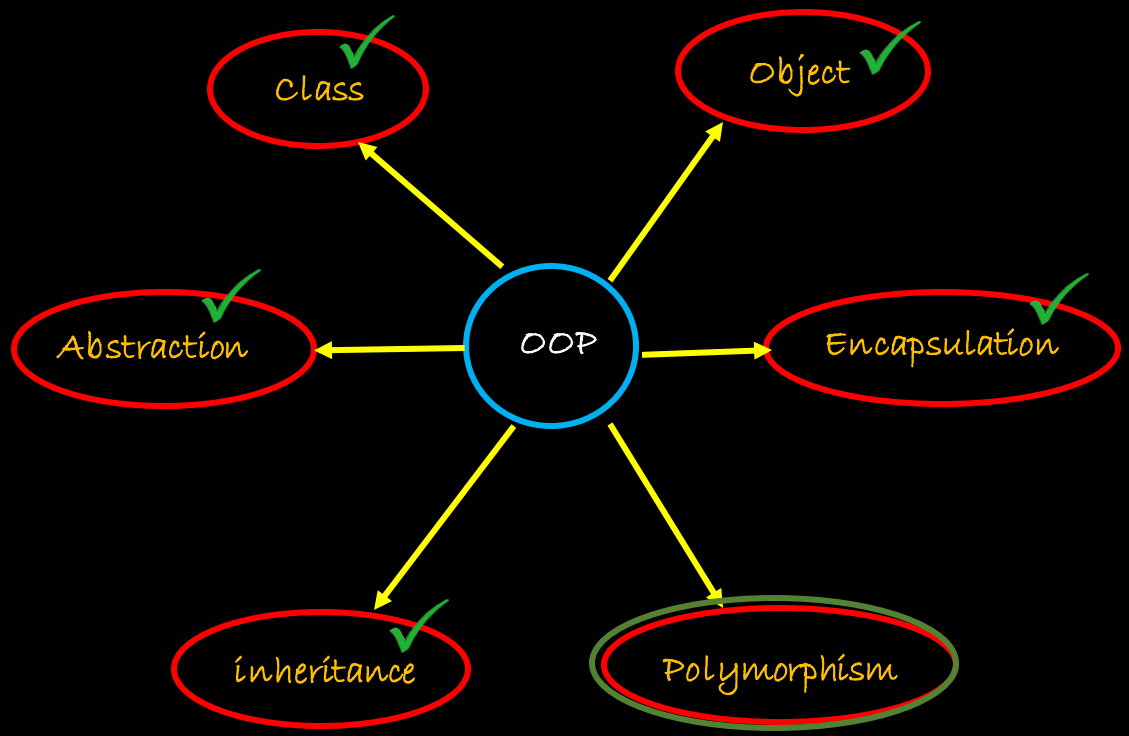
Lecture 14
Polymorphism in C++

By
Ritwik M
Assistant Professor(SrGr)
Dept. Of Computer Science & Engg
Amrita Vishwa Vidyapeetham - Coimbatore

Based on material from Stroustrup B. Programming: principles and practice using C++. Second edition,Addison Wesley; 2014 and multiple sources on the internet

A Quick Recap

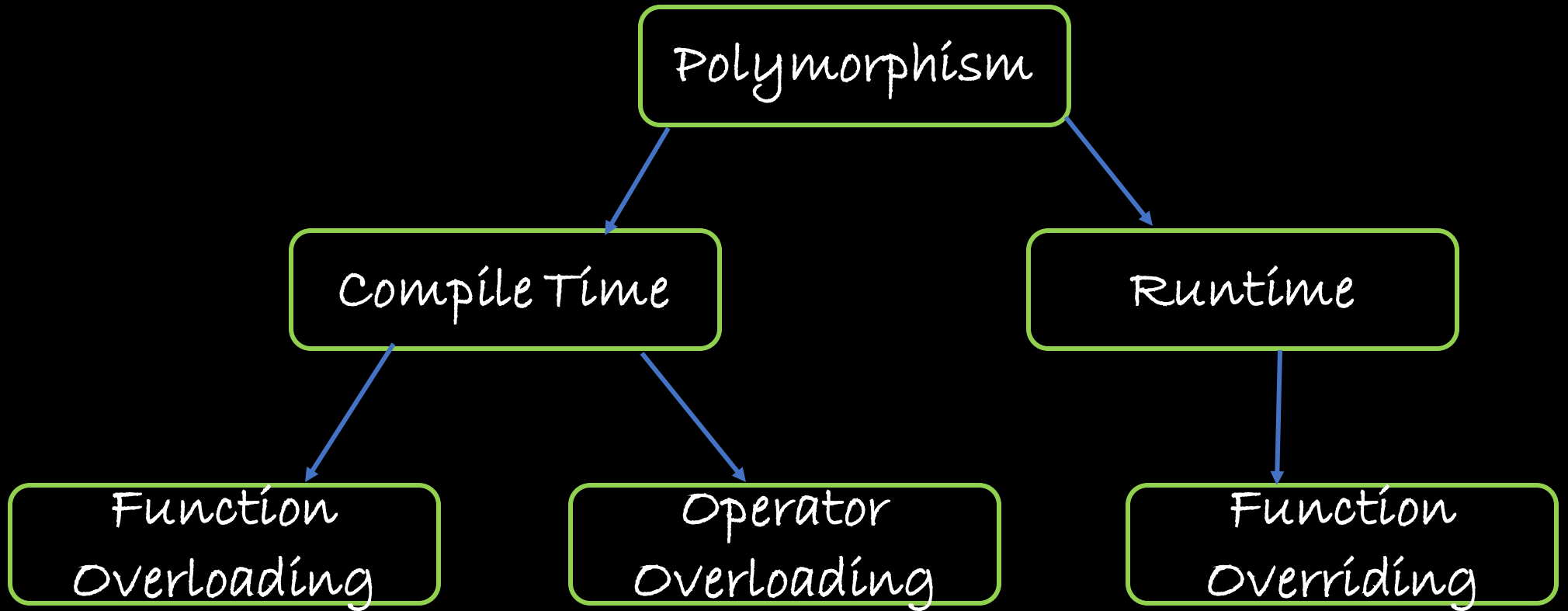
- Inheritance
- Memory Management in C++
- New & Delete keywords
- Examples and Exercises



Polymorphism

- Literal Meaning
  - Poly – many
  - Morph - denoting something having a specified form or character.
- Polymorphism
  - the condition of occurring in several different forms.
- Real World Example:
  - A person has different characteristics – the child, the friend, the student, etc.

Types


Polymorphism
Compile Time
Runtime
Function Overloading
Operator Overloading
Function Overriding

Compile Time Polymorphism - Overloading

- Used to avoid redundant code where the same method name or operator is used multiple times but with a different set of parameters or number of operands
- The actual method that gets called during runtime is resolved at compile time, thus avoiding runtime errors
- Overloading provides code clarity; reduce complexity, and increases runtime presentation of a code

Function Overloading

- The process of having two or more member functions of a class with the same name, but different in parameters.
- Strategy
  - Changing number of arguments.
  - Changing the data type of arguments.
- Advantage
  - It increases the readability of the program because we don't need to use different names for the same action again and again.
- Well known example:
  - Default constructor Vs Parameterized constructor

Operator Overloading

- Most of the built-in operators available in C++ can be redefined
- Operator overloading provides a flexibility option for creating new definitions of C++ operators.
- Syntax


```
returnType  operator op(arg_list)
 {//Function Body}
```

- Example

```
// overloaded minus (-) operator
      Distance operator- () {
         feet = -feet;
         return Distance(feet);
      }
```

Operator Overloading - Rules

- Precedence will not change due to overloading.
- Associativity will not get effected.
- Commutativity Overloading cannot change the commutativity of the operator.
- Arity If a native operator is unary (taking only one operand), then the overloaded definition must also be unary.
- No new operators We cannot invent operators.
- No combination We cannot combine two operator symbols to create a new one.

More on Operator Overloading

- Operators can be overloaded as class member functions or as friend functions.
- The signature of the operator function differs based on
  - The type of operator (unary or binary operator) overloaded
  - Type of function (member of friend function) used for overloading.
- Member functions are preferred when the host object is to be modified.
  - Example: obj++, -obj (unary -), obj1+=obj2, obj1=obj2.
- NOTE: the >> and << operators can be overloaded only as friend functions

More on Operator Overloading

- Unary Operators Overloading
- Binary Operators Overloading
- Relational Operators Overloading
- Input/Output Operators Overloading
- ++ and -- Operators Overloading
- Assignment Operators Overloading
- Function call () Operator Overloading
- Subscripting [] Operator Overloading
- Class Member Access Operator -> Overloading

Operator Overloading : Example

```
class Distance {
   private:
      int feet; // 0 to infinite
      int inches;    // 0 to 12
   public:
  // required constructors
      Distance() {
         feet = 0;
         inches = 0;
      }
```

1

```
      Distance(int f, int i) {
         feet = f;
         inches = i;
      }
   // method to display distance
   void displayDistance() {
   cout << "F: " << feet <<  " I:" <<  inches;
   }
```

2

```
//operator overloading
 void operator + (const Distance &D ){
     feet = feet+D.feet;
     inches = inches+D.inches;
   }
};
```

3

```
//Main function
int mainwhat is () {
   Distance D1(11, 10), D2(5, 11);
    cout << "First Distance : "<<endl;
    D1.displayDistance();
   cout << "Second Distance :"; D2.displayDistance();
   // use overloaded assignment operator
   D1 + D2;
   cout << "First Distance :"; D1.displayDistance();
   return 0;
}
```

4

Note on Operator Overloading

- Operators that cannot be overloaded
  - Class member access operator (. (dot), .* (dot-asterisk))
  - Scope resolution operator (::)
  - Conditional Operator (?:)
  - Size Operator (sizeof)

Exercise - 1

- Write a C++ program to overload the relational operators.
- Write a C++ program to evaluate the following expression "Obj1+ Obj2 - Obj3"

Exercise 2

- Given the following function declaration, find the error in the following

1.
```
int max(int);
float max(int);
```
   - Answer: Functions that differs only by return type cannot be overloaded.

- 2.
```
void sum(int);
void sum(float);
void sum(char);
```
   - Answer: It leads to ambiguity for a function call sum(65)' as all the three function signature matches with call because of type conversion.

Quick Summary

- Polymorphism
- Compile-time Polymorphism
- Runtime polymorphism
- Function Overloading
- Operator Overloading
- Examples
- Exercises

Up Next

Runtime Polymorphism in C++