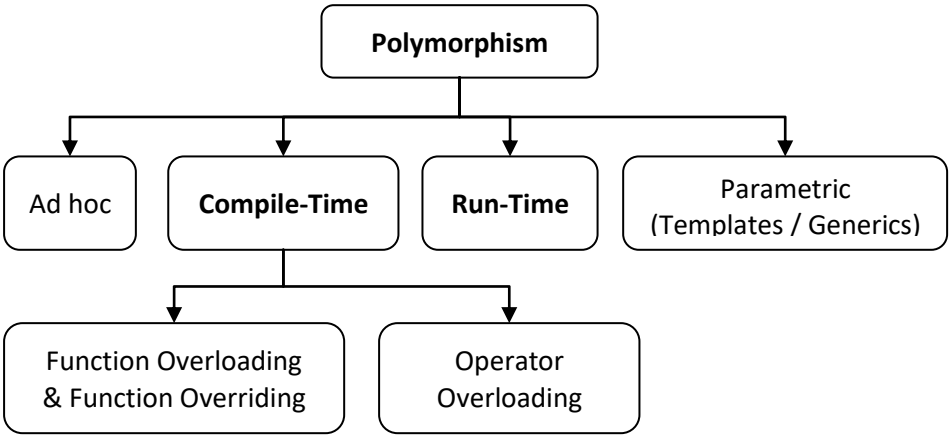
Unit 5: Polymorphism

5.1 Polymorphism in Programming Language

- The word Polymorphism is derived from Greek words “poly” meaning many and “morphos” meaning forms.
- Polymorphism in object oriented programming language is an important feature/concept which refers to the ability of a variable, function or object to take on multiple forms and show a different set of behavior depending upon the context.
- It allows a programmer to write code in a more general way rather than being specific. For example, the same function name Display() may be used to print out the information of different objects belonging to a class as well as objects of different classes.
- The C++ language itself implements polymorphism features in the core of the language. For example, the same ‘+’ operator can be used to add two numeric data as well as concatenate two strings.

5.2 Varieties of Polymorphism



5.2.1 Compile-time Polymorphism

If the polymorphism feature is implemented such that the compiler is able to decide which of the alternate functions to use during the compilation of source code into binary executable, it is known as compile-time polymorphism. It is also referred to as early binding or static binding.

Function Overloading

When the function with same name is defined with multiple versions in the same class or derived class(es) by varying the number and/or types of arguments, it is known a function overloading.

Function Overriding

When a function with same name and type and number of arguments from the base class is redefined in a derived class in an inheritance hierarchy, it is known as function overriding.

Operator Overloading

It is the process of redefining or extending the meaning of different operators like +, – , etc. which are already defined for basic data types like int, float, etc. so that they work equally well with user defined data types (classes). It allows us to do such basic operations in a more intuitive way for user defined types.

For example, suppose we need to store the sum of two complex numbers C1 and C2 into C3. In this case, instead of something like `C3.add(C1, C2)`, it would be more meaningful and easier to use `C3 = C1 + C2`. Here, we have to overload the '+' operator for using the later expression.

Data Type Conversion

The process of making provision for converting one data type to another compatible data type just using the = operator is known as data type conversion. For data type conversion both the source and destination data types may be either basic data type like int, float, etc. or user defined type (class). The conversion between two basic data types is done automatically by the compiler and is known as

coercion or typecasting. On the other hand, the code for conversion between two user-defined classes or between basic type and user-defined class should be written by the programmer.

5.2.2 Run-time Polymorphism

If the polymorphism feature is implemented such that which of the alternate polymorphic function to use is decided only during the execution of the program, it is known as run-time polymorphism. It is also referred to as late binding or dynamic binding. In C++, it is achieved by defining virtual function(s) in the base class and implementing the virtual function(s) in derived class(es) allowing the pointer of base class to be used for the object(s) of derived class(es) such that appropriate child class methods are automatically invoked even through the base class pointer.

Polymorphic Variable

A polymorphic variable is a variable that can reference more than one type of object during the course of program execution. Polymorphic variables derive their power from interaction with inheritance, overriding and substitution principle.

Object Pointer

Like basic data types, we can also create a pointer of a class and let it point to an object of the same class. Additionally, we can also use the base class pointer to point to a derived class object.

this Pointer

Every object in C++ has access to its own address through an important pointer called this pointer. The ‘this’ pointer is an implicit parameter to all member functions. Therefore, inside a member function, the ‘this’ pointer may be used to refer to the invoking object. Especially, it becomes useful when the local variables or arguments of a member function have the same name as that of class members.

Virtual Function

A virtual function is a member function in the base class that we expect to redefine in derived classes. Basically, a virtual function is used in the base class in order to ensure that the function is overridden. This especially applies to cases where a pointer of base class points to an object of a derived class and one needs to invoke a derived class method through the base class pointer.

5.2.3 Ad hoc Polymorphism

The encapsulation feature of OOP allows us to bundle together the related properties and methods as data members and member functions into a single entity called class from which we can create or instantiate any number of objects. When we invoke a method for a particular object, the compiler is able to distinguish the function call from rest of the objects. Furthermore, it is common to have the methods with same function name across different classes. Invoking such functions for a particular object of a particular class can be done without any ambiguity. This is known as ad hoc polymorphism.

5.2.4 Parametric Polymorphism

It is the process of designing functions or classes in such a way that the same body of code works equally well with various data types. This is implemented in C++ using template functions or template classes and is also known as generics. Simply put, when we make the data type used in a function or a class vary across implementations such that the data type itself is parameterized, this is known as parametric polymorphism.