Classes in C++

Classes are used to group together data and methods, called members.

A member can be static (with the **static** keyword) or not (no keyword).

By default, members are not static (and are thus instance members). In this case, they can only be accessed through a class instance. To access a member of an instance, we write <instance>●<member>. In an instance method, the **this** pointer contains the address of the instance from which the method is called. Attributes of an instance are stored in the allocated memory of the instance.

A static member (also known as a class member) can be used without there being any instance of the class. A static member can be accessed through an instance (*<instance>●<member>)* or by class name (<class>**::**<member>).

There is no **this** pointer in a static method.

Static data exists as a single copy in memory.

Objects of classes consist of instance data in memory, and can execute instance methods or static methods of their class.

**class** grouping (encapsulation) of data (properties) and code (methods).

**instance of a class** object of type class that occupies a memory block that holds the instance data.

**attribute or property** data member.

**method** member function.

**instance member** member that can only be accessed through an instance of the class.

**static member** member that exists with or without instances.

**access modifier** specifies the visibility or accessibility of a member (data or function):

- public : visible from all methods (functions)

- protected : visible only to methods in the class or in derived classes

- private : visible only to the methods of this class

**accessor (or getter)** method that returns the value of a property

**setter** method that takes a value as a parameter and stores it in a property.

**constructor** special method that is automatically called when a new instance is created. If no constructor is defined, the compiler will create one called the default constructor. The default constructor has no parameters and does nothing.

use: AClass object( <parameters> );

declaration: AClass( <list of formal parameters> );

definition: AClass::AClass( <list of formal parameters> ) { <body> }

**copy constructor** constructor that allows you to initialize an object for the first time as a copy of another object of the same type.

By default, this method copies the memory block of the object passed as a parameter into that of the object being created.

uses: AClass obj1( obj2 ); or AClass obj1 = obj2;

passing by value in a function call

returning an object by function value

declaration: AClass( const AClass & );

definition: AClass::AClass( const AClass & ) { <body> }

**assignment operator** used to initialize an object that has already been created with another object of the same type. By default, this method copies the memory block of the object on the right into that of the object on the left.

use: AClass obj1, obj2;

obj1 = obj2;

declaration: const AClass &operator=( const AClass & );

definition: const AClass & AClass::operator=( const AClass & ) { <body> }

**destructor** special method called automatically right before an object is deleted. By default, the destructor does nothing.

declaration: ~ AClass ( );

definition: AClass:: ~ AClass ( ) { <body of the method> }

For best control over object properties, all data members should be private.