Access Specifiers

Access specifiers define how the members (attributes and methods) of a class can be accessed.

Types of Access specifiers

In C++, there are three access specifiers:

- public - members are accessible from outside the class
- private - members cannot be accessed (or viewed) from outside the class
- protected - members cannot be accessed from outside the class, however, they can be accessed in inherited classes. We'll go through with it in Inheritance topic.

Example1: accessing variables from public and private,

```
class new_Class {

  public:

    int public_var = 10;

  private:

    int private_var = 20;

};


int main(void) {

  new_Class obj;

  cout << "Access public variable" << obj.public_var << endl;

  cout << "Access private variable" << obj.private_var;

}
```

Example2: accessing methods from public and private,

```
class new_Class {
   public:
      int public_method(){
         return 10;
      }
   private:
      int private_method(){
         return 20;
      }
};

int main(void) {
   new_Class obj;
   int public_var = obj.public_method();
   int private_var = obj.private_method();
   cout << "Access public method" << public_method << endl;
   cout << "Access private method" << private_var;
}
```

Note: By default, all members of a class are private if you don't specify an access specifier:

Example:

```
class new_Class {
      int var1 = 10;          // Private attribute
      int var2 = 20;          // Private attribute
};
```

objects and memory

There are two parts of memory in which an object can be stored:

- stack – Memory from the stack is used by all the members which are declared inside blocks/functions. Note that the main is also a function.

- heap – This memory is unused and can be used to dynamically allocate the memory at runtime.

The scope of the object created inside a block or a function is limited to the block in which it is created.

- The object created inside the block will be stored in the stack and Object is destroyed and removed from the stack when the function/block exits.

- But if we create the object at runtime i.e by dynamic memory allocation then the object will be stored on the heap. This is done with the help of new operator. In this case, we need to explicitly destroy the object using delete operator.