Object Oriented Programming Using C++

Ketan G Kore
Ketan.kore@sunbeaminfo.com

Storage Classes

- Following are the storage classes in C/C++:
    1. **auto**
    2. **static**
    3. **extern**
    4. **Register**

- Storage class decides scope of variable and function and lifetime of the variable.

Scope in C++

- Scope decides region/area/boundary where we can access variable/function.
- Following are the scope in C++:
    1. **Block scope**
    2. **Function scope**
    3. **Prototype scope**
    4. **Class Scope**
    5. **Namespace Scope**
    6. **File Scope**
    7. **Program Scope**

Lifetime in C++

- Lifetime represents existence of variable/object inside RAM.

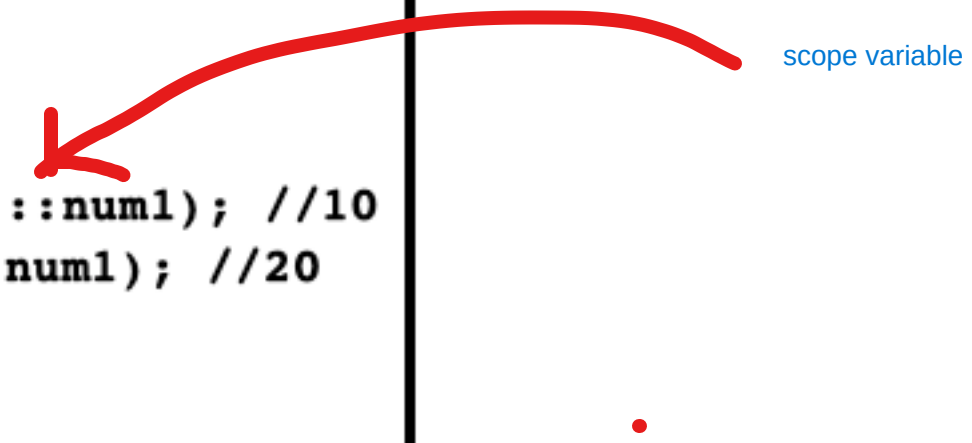
- Following are the lifetimes in C++:

    1. **Automatic lifetime**
    2. **Static lifetime**
    3. **Dynamic lifetime**

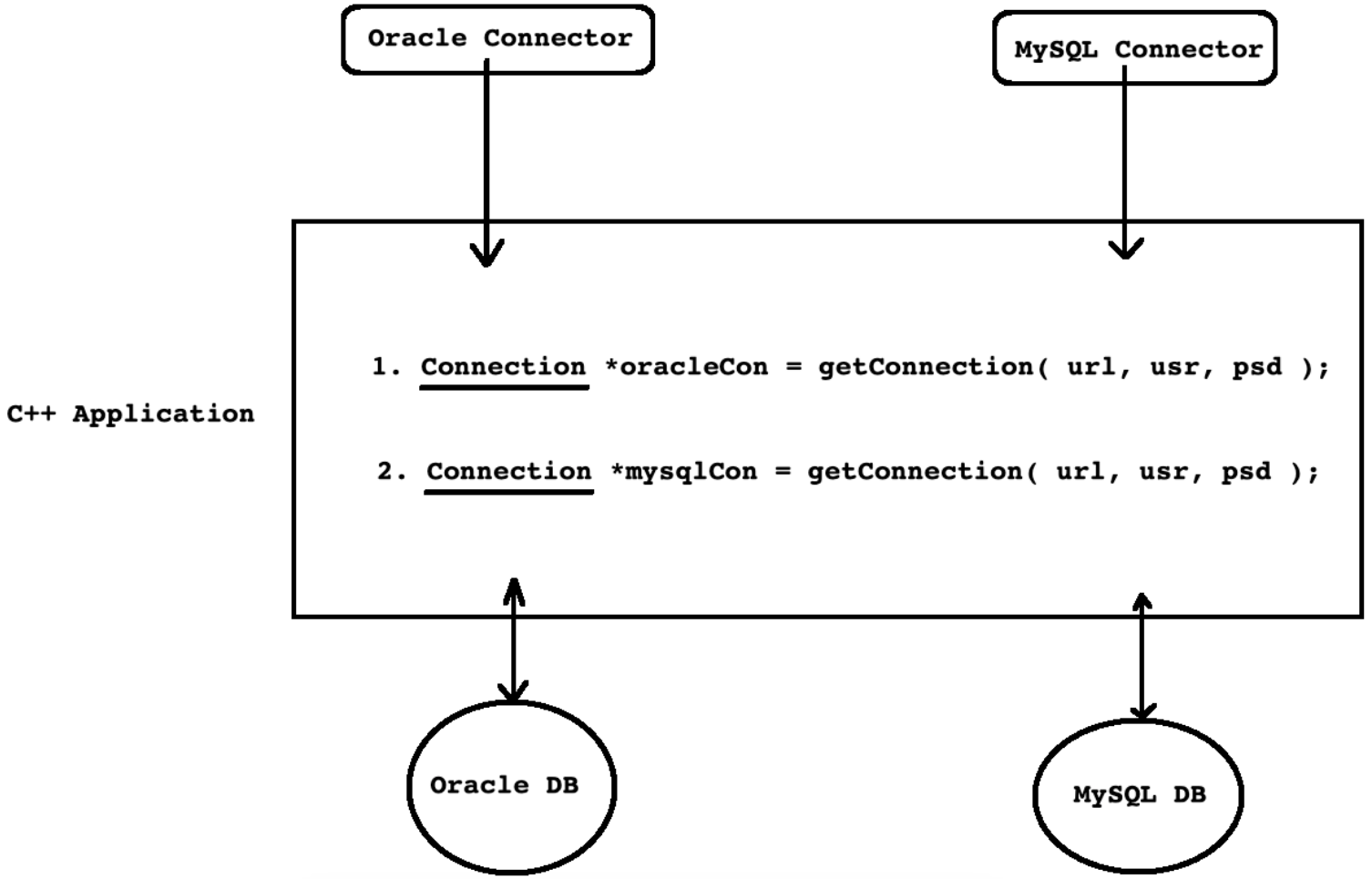
Scope Resolution Operator(::)

- Inside same scope, we can not give same name to the multiple variables. But name of local variable and global variable can be same.
- If name of local variable and global variable is same then preference is always given to the local variable.
- In C++, using scope resolution operator, we can access value of global variable inside function.

```
int num1 = 10;
int main( void ){
   int num1 = 20;
   printf("Num1 :  %d\n", ::num1); //10
   printf("Num1 :  %d\n", num1); //20
   return 0;
}
```



Need Of Namespace



Namespace

- Namespace is C++ language feature that we can use:
    1. To avoid name clashing/collision/ambiguity.
    2. To group functionally related / equivalent types together
- Using namespace keyword, we can define namespace. Consider following code.

```
namespace na{
   int num1 = 10;
}
int main( void ){
   printf("Num1 :  %d\n", na::num1);
   return 0;
}
```

- To access members of namespace, we should use :: operator.

Namespace

- If name of the namespaces are different then name of members of namespace may/may not be same. Consider following code.

```
namespace na{
   int num1 = 10;
   int num3 = 30;
}
namespace nb{
   int num2 = 20;
   int num3 = 40;
}
int main( void ){
   printf("Num1 :  %d\n", na::num1); //10
   printf("Num3 :  %d\n", na::num3); //30

   printf("Num2 :  %d\n", nb::num2); //20
   printf("Num3 :  %d\n", nb::num3); //40
   return 0;
}
```

Namespace

- If name of the namespaces are same then name of members of namespace can not be same. Consider following code.

```
namespace na{
   int num1 = 10;  //OK
   int num3 = 30;  //OK
}
namespace na{
   int num2 = 20;  //OK
   int num3 = 40;  //Not OK
}
int main( void ){
   printf("Num1 :  %d\n", na::num1); //10
   printf("Num3 :  %d\n", na::num3); //30

   printf("Num2 :  %d\n", na::num2); //20
   return 0;
}
```

Nested Namespace

- We can define namespace inside another namespace. It is called nested namespace.

```
int num1 = 10;
namespace na{
   int num2 = 20;
   namespace nb{
      int num3 = 30;
   }
}
int main( void ){
   printf("Num1 :  %d\n", ::num1); //10
   printf("Num2 :  %d\n", na::num2); //20
   printf("Num3 :  %d\n", na::nb::num3); //30
   return 0;
}
```

- Global members are considered as a member of global namespace.
- To access members of global namespace, we can use only :: operator.

using directive

- If we want to use members of namespace frequently then we should use using directive.

```
namespace na{
   int num1 = 10;
}
int main( void ){
   using namespace na;
   printf("Num1 :  %d\n", num1); //10
   return 0;
}
```

```
namespace na{
   int num1 = 10;
}
int main( void ){
   int num1 = 20;
   using namespace na;
   printf("Num1 :  %d\n", num1); //20
   printf("Num1 :  %d\n", na::num1); //10
   return 0;
}
```

using directive

- If we want to use members of namespace in multiple function then we should define using directive global.

```
namespace na{
   int num1 = 10;
}
using namespace na;
void show_record( void ){
   printf("Num1 :  %d\n", num1);
}
void display_record( void ){
   printf("Num1 :  %d\n", num1);
}
int main( void ){
   ::show_record( );
   ::display_record( );
   return 0;
}
```

here we are using global namespace

Points to remember about namespace

- We can give same /different name to the namespaces.
- To access members of namespace either we should use :: operator / using directive.
- We can not define namespace inside function/class. It should be global / it should be inside another namespace.
- We can not define main function inside namespace. It must be member of global namespace.
- We can not create object of namespace. It is used for grouping.
- std is standard namespace in C++.
- Generally name of the namespace should be in lower case.

Standard Streams Of C++

- Stream is a an abstraction(object) which is used to produce(write) and consume(read) data from source to destination.
- Console = Keyboard + Monitor.
- Following are the standard streams of C++ that is associated with console:
    1. cin : input stream represents keyboard
    2. cout : output stream represents monitor
    3. cerr : **Unbuffered** standard error stream
    4. clog : **Buffered** standard error stream

Standard Streams Of C++

Header File : <iostream>

```
namepace std{
    extern istream cin;
    extern ostream cout;
    extern ostream cerr; //UnBuffered
    extern ostream clog; //Buffered
}
```

Character Output(cout)

- cout stands for character output.
- It is standard output stream of C++ which represents monitor. In other words, if we want to print state of object on console/monitor then we should use cout.
- cout is object of ostream class and ostream is nickname/alias of basic_ostream<char> class.

  typedef basic_ostream<char> ostream;

- cout is declared as extern object inside std namespace (hence std::cout) and std namespace is declared in <iostream> header file.
- cout is not a function hence to read data from keyboard we must use insertion(<<) operator with it.

Character Output(cout)

```
#include<iostream>
int main( void ){
   int num1 = 10;
   std::cout << num1 << std::endl;
   return 0;
}
```

1

```
#include<iostream>
int main( void ){
   using namespace std;
   int num1 = 10;
   cout << num1 << endl;
   return 0;
}
```

2

```
#include<iostream>
int main( void ){
   using namespace std;
   int num1 = 10 num2 = 20;
   cout << num1 << num2 << endl;
   return 0;
}
```

3

```
#include<iostream>
int main( void ){
   using namespace std;

   int num1 = 10;
   cout << "Num1   :  " << num1 <<endl;

   int num2 = 20;
   cout << "Num1   :  " << num1 <<endl;

   return 0;
}
```

4

Character Input(cin)

- cin stands for character input.
- It is standard input stream of C++ which represents keyboard. In other words, if we want to read data from console/keyboard then we should use cin.
- cin is object of istream class and istream is nickname/alias of basic_istream<char> class.

  typedef basic_istream<char> istream;

- cin is declared as extern object inside std namespace (hence std::cin) and std namespace is declared in <iostream> header file.
- cin is not a function hence to read data from keyboard we must use extraction(>>) operator with it.

Character Input(cin)

```
#include<iostream>
int main( void ){
   int num1;
   std::cin >> num1;
   //TODO : print num1;
   return 0;
}
```

1

```
#include<iostream>
int main( void ){
   using namespace std;
   int num1;
   cin >> num1;
   //TODO : print num1;
   return 0;
}
```

2

```
#include<iostream>
int main( void ){
   using namespace std;
   int num1, num2;
   cin >> num1 >> num2;
   //TODO : print num1 and num2;
   return 0;
}
```

3

```
#include<iostream>
int main( void ){
   using namespace std;

   int num1;
   cin >> num1;

   int num2;
   cin >> num2;

   //TODO : print num1 and num2;
   return 0;
}
```

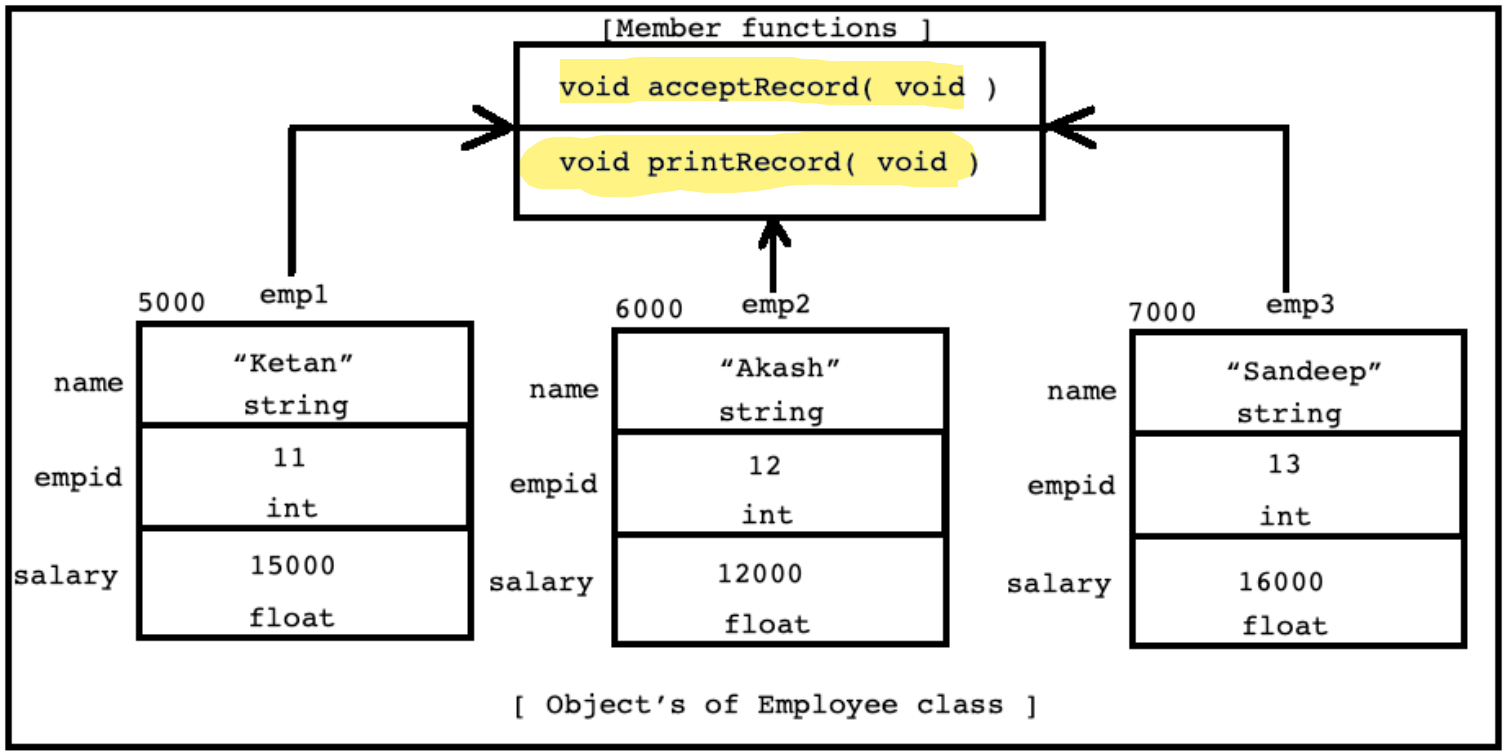
4

Manipulator

- In C/C++, escape sequence is a character which is used to format the output.
- Example : ‘\n’, ‘\t’, ‘\b’, ‘\a’ etc.
- In C++, manipulator is a function which is used to format the output.
- Example : endl, setw, fixed, scientific, setprecision, dec, oct, hex etc.
- All the manipulators are declared in std namespace but header files are different
    1. To use endl include <iostream> header file
    2. To use remaining manipulators include <iomanip> header file
- Reference : https://en.cppreference.com/w/cpp/io/manip

Object Oriented Concepts.

- Data members of the class get space once per object according their order of declaration inside class.
- Member function do not get space inside object, rather all the objects of same class share single copy of it.



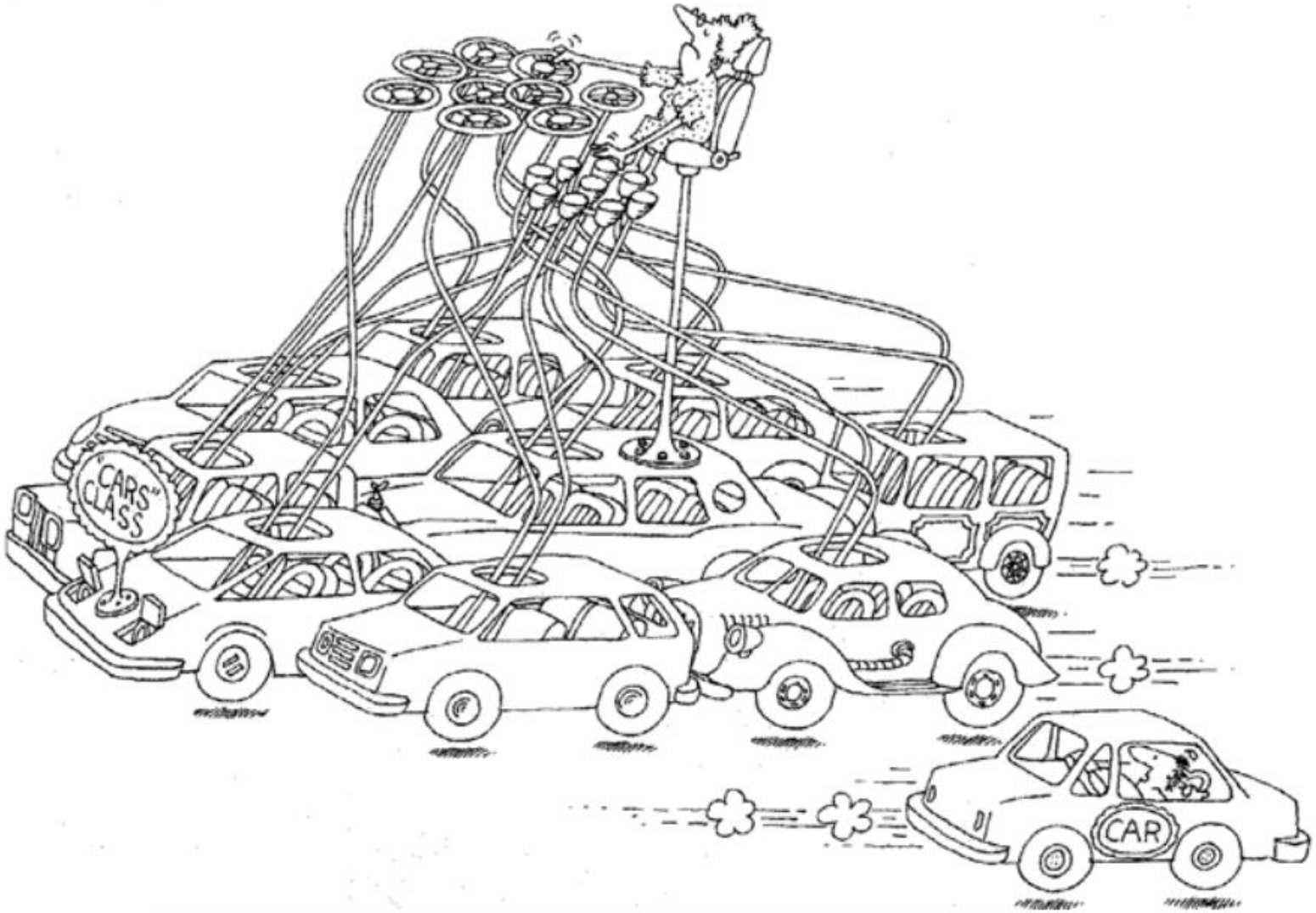
Object Oriented Concepts.

- **Characteristics of Object**
    1. **State**
        - Value/data stored inside object is called state of the object.
        - Value of the data member represent state of object.
    2. **Behavior**
        - Set of operations that we can perform on object is called behavior of the object.
        - Member functions defined inside class represents behavior of the object.
    3. **Identity**
        - Identity is that property of an object which distinguishes it from all other objects.
- Empty class
    - A class which do not contain any member is called empty class.
    - **Size of object of empty class is 1 byte**.

Class

- Definition:
  - A class is collection of data member and member function.
  - Structure and behavior of the object depends on class hence class is considered as a template/model/blueprint for object.
  - A class represents a set of objects that share a common structure and a common behavior.
- Class is logical or imaginary entity.
- Example
  - Car
  - Book
  - Mobile Phone
- Class implementation represents encapsulation.

Class



Object

- Definition:
  - Object is a variable / instance of class.
  - Any entity which is having physical existence is called object. In other words, an entity which get space inside memory is called object.
  - An entity, which has sate, behavior and identity is called object.
- Object is physical or real time entity.
- Example
  - Tata nano
  - C++ Complete reference
  - Nokia 1100

this pointer

- **Consider steps in program development**

    1. Understand clients requirement i.e problem statement.
    2. Depending on the requirement, define class and declare data member inside it.
    3. Create object of the class.(Here data member will get space inside object)
    4. To process state of the object, call member function on object and define member function inside class.

- If we call member function on object then compiler implicitly pass, address of current object as a argument to the function.
- To store address of argument(current object), compiler implicitly declare one pointer as function parameter inside member function. It is called this pointer.
- General type of this pointer is : **ClassName *const this;**
- **this** is a keyword in C++.

this pointer

- Using this pointer, non static data member and non static member function can communicate with each other hence, it is considered as a link/connection between them.
- If name of the data member and name of local variable is same then preference is always given to the local variable. In this case to refer data member, we must use this keyword. Otherwise use of this keyword is optional.
- this pointer is implicit pointer, which is available in every non static member function of the class which is used to store address of current object or calling object.
- Following member function do not get this pointer:
    1. Global function
    2. Static member function
    3. Friend function
- We can not declare this pointer explicitly. But compiler consider this pointer as a first parameter inside member function.

Thank you