C++ for C developers

Tarek Ibrahim Eltorky

Contents

1. What is Class and Object

Class:

A class in C++ is the building block, that leads to Object-Oriented programming. It is a user-defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A C++ class is like a blueprint for an object.

structure/class: Security”public/private” “Cannot be inherited/Can” “Doesn’t contain methods/it does”

Object :

is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.

2. Some differences between C and C++

C	C++
C cannot run C++ code.	C++ can run most of C code.
supports procedural programming paradigm for code development.	supports both procedural and object oriented programming paradigms, it is also called a hybrid language.
being a procedural programming, it is a function driven language. And it does not support OOP principles.	Being an object oriented programming language C++ supports abstraction, polymorphism, encapsulation, and inheritance.
does not support function and operator overloading	support function and operator overloading

C	C++
does not support reference variables.	support reference variables.
provides malloc() and calloc() functions for dynamic memory allocation, and free() for memory de-allocation.	provides new operator for memory allocation and delete operator for memory de-allocation.
Does not provide direct support for error handling (also called exception handling)	provides support for exception handling. And it is a mechanism allows you to take appropriate action to avoid runtime errors.
does not allow functions to be defined inside structures.	functions can be used inside structures and classes.
uses functions for input/output. ex.scanf and printf.	uses objects for input output, e.x. cin and cout

3. Enumerations and type safety

```c

#include <stdio.h>


int main()
{
    char* b ="aaa";
    printf("%s %d",b,(int)b);

    return 0;
}

```

//C is not type safe

input

```
main.c:15:22: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
aaa 4195828

...Program finished with exit code 0
Press ENTER to exit console.
```

```
8
9  #include <iostream>
10
11
12 int main()
13 {
14     std::string b ="aaa";
15     std::cout << b << static_cast<int>(b) << std::endl;
16
17     return 0;
18 }
19
```

//C++ is type safe

input

Compilation failed due to following error(s).

```
main.cpp: In function ‘int main()’:
main.cpp:15:41: error: invalid static_cast from type ‘std::string {aka std::basic_string}’ to type ‘int’
     std::cout << b << static_cast<int>(b) << std::endl;
                                        ^
```

Enumerations

In C ,there is no variable can have a name which is already in some enumeration as shown in this example :

- YELLOW cannot be in different enums

In C++ there is an enum class which can holds similar contents with different enum class names

- that makes enumerations both strongly typed and strongly scoped.
- Class enum doesn't allow implicit conversion to int, and also doesn't compare enumerators from different enumerations.

https://onlinegdb.com/H1a5EeD58

```
9  #include <stdio.h>
10
11 enum color
12 {
13     RED,
14     GREEN,
15     YELLOW
16 };
17
18 enum stoplight
19 {
20     WE,
21     YELLOW,
22     TES
23 };
24
25 int main()
26 {
27     enum color x = YELLOW;
28     printf("Hello %d",x);
29
30     return 0;
31 }
32
```

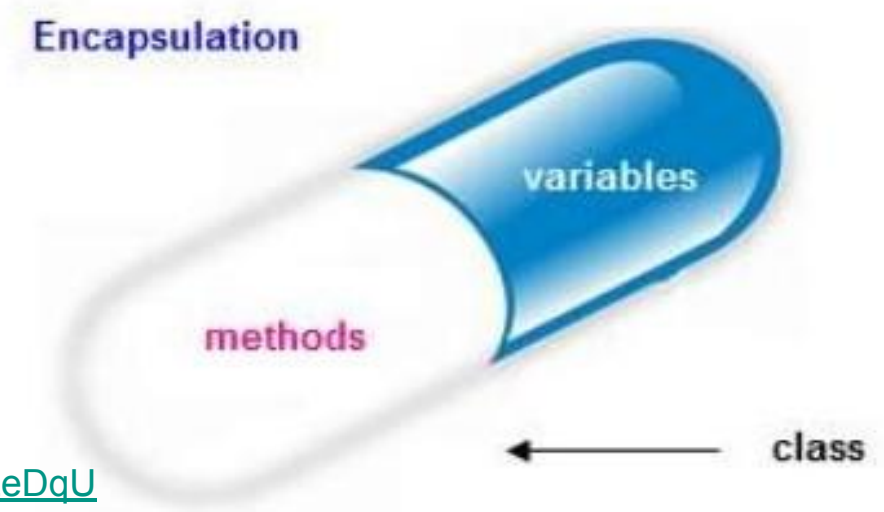
input

Compilation failed due to following error(s).

```
main.c:21:5: error: redeclaration of enumerator 'YELLOW'
     YELLOW,
     ^~~~~~
main.c:15:5: note: previous definition of 'YELLOW' was here
     YELLOW
     ^~~~~~
```

4. OOP Principles in C++

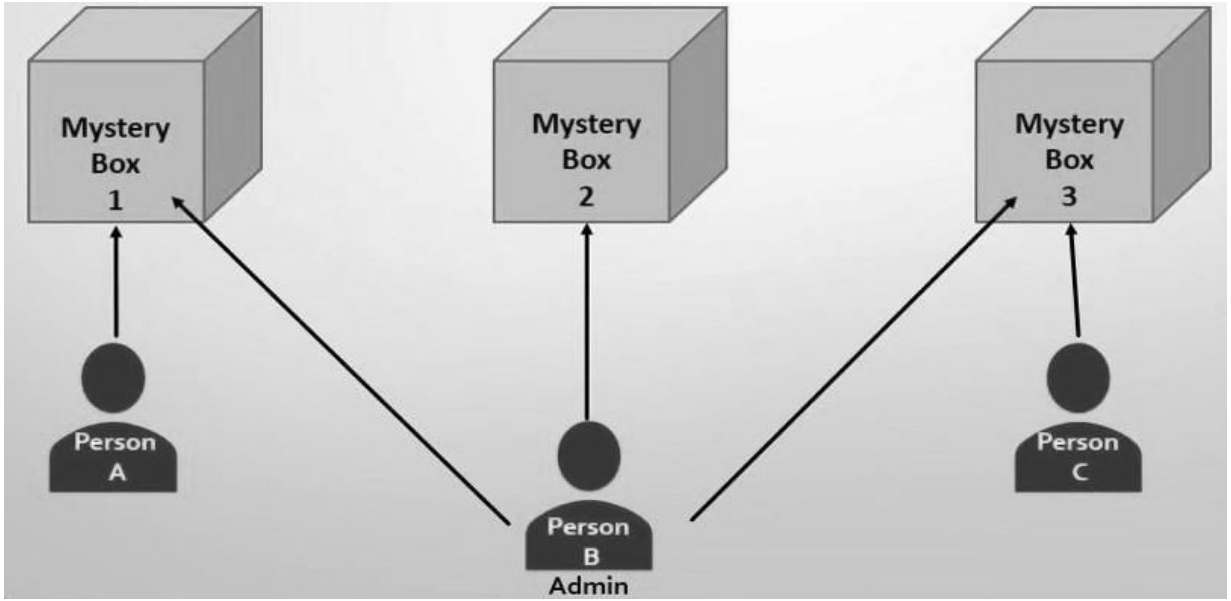
Encapsulation
is defined as wrapping up of data and information under a single unit. In Object Oriented Programming, Encapsulation is defined as binding together the data and the functions that manipulates them.



https://onlinegdb.com/Sy0BLeDqU

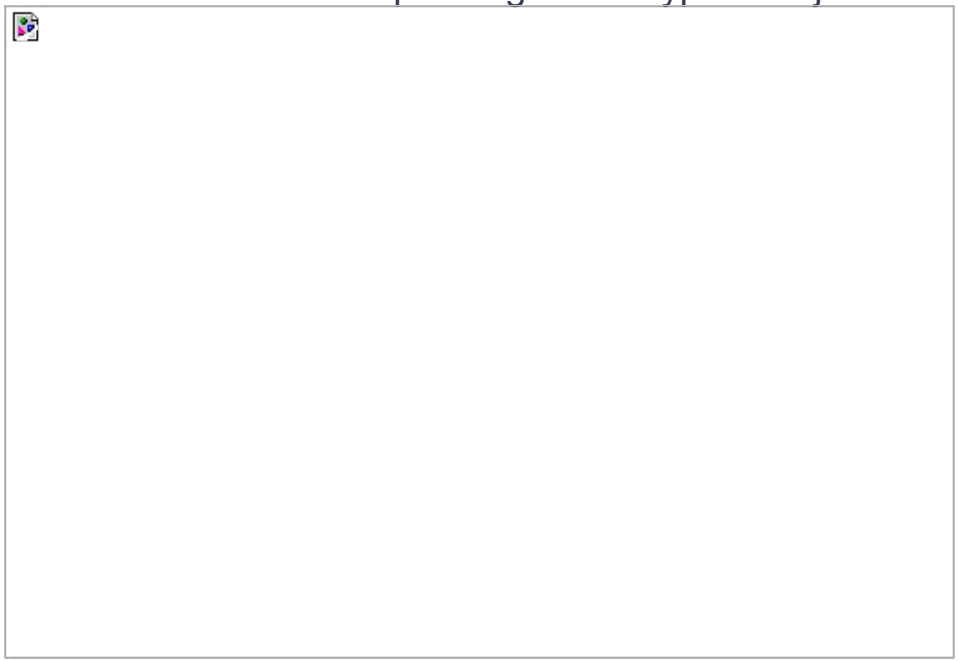
Abstraction

means displaying only essential information and hiding the details. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation.



Polymorphism

The word polymorphism means having many forms. It means that a call to a member function will cause a different function to be executed depending on the type of object that invokes the function.



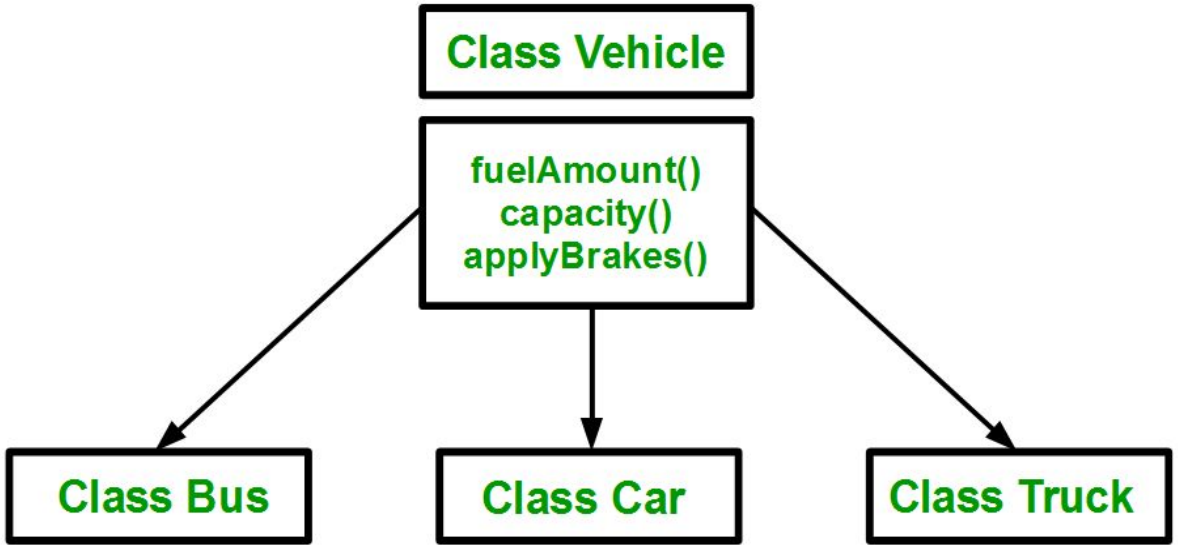
https://onlinegdb.com/Syv2dYzoU

Inheritance

The capability of a class to derive properties and characteristics from another class is called Inheritance.

Sub Class: The class that inherits properties from another class is called Subclass or Derived Class.

Super Class:The class whose properties are inherited by subclass is called Base Class or Super class.



Quiz

1. Make a class called Animal
   name and age , setter and getter to both
2. Make a subclass called Dog
   + owner, set it and get it
3. Make a function called foo() in the base class and override it in the subclass

solution: https://onlinegdb.com/HJ1_o-_qU

5. Constructors and Destructors

Constructor

It is a member function of a class which initializes objects of a class. In C++, Constructor is automatically called when object(instance of class) create. It is special member function of the class.

- Constructor has same name as the class itself
- Constructors don’t have return type
- A constructor is automatically called when an object is created.
- If we do not specify a constructor, C++ compiler generates a default constructor for us (expects no parameters and has an empty body).

“First thing to be called when an object is created”

Destructor

is a member function which destructs or deletes an object.there can only one destructor in a class with class name preceded by “tilda”~, no parameters and no return type.

When is destructor called?

A destructor function is called automatically when the object goes out of scope:

- the function ends
- the program ends
- a block containing local variables ends
- a delete operator is called

“Last thing to be called at the end of an object life”

https://onlinegdb.com/B18uDni98

6. Generics in C++ <Template>

The simple idea is to pass data type as a parameter so that we don’t need to write the same code for different data types.
Templates are expanded at compilation time. This is like macros.

- The difference is, compiler does type checking before template expansion.

The idea is simple, source code contains only function/class, but compiled code may contain multiple copies of same function/class.

- Macro cannot check the data type of arguments
- Macros are expanded by the preprocessor and then compilation takes place. Compiler will refer error messages in expanded macro or the line where macro has been called.

The template idea is to make a generic code that can be used for more than one purpose as shown in the example :

The function sum() takes 2 or 3 args If the 3rd arg is not passed so it is zero by default.

sum function will accept any data type which is subject to arithmetic operations as its implemented.

Another example on template :
https://onlinegdb.com/r1p9sdF9U

```cpp
#include <iostream>


template <class T>
inline T sum(T* arr,int size,T sum=0)
{
    for(int i=0;i<size;i++)
    {
        sum += arr[i];
    }
    return sum;
}

int main(void)
{
    double arr_doub[5]={1.1,2.2,3.3,4.4,5.5};
    int arr_int[5]={1,2,3,4,5};
    std::cout << sum(arr_doub,5) << std::endl;
    std::cout << sum(arr_int,5) << std::endl;
    return 0;
}
```

```
16.5
15
```

7. Vectors in STL "Standard Template Library"

- Just like arrays, vectors use contiguous storage locations for their elements, which means that their elements can also be accessed using offsets on regular pointers to its elements, and just as efficiently as in arrays. But unlike arrays,they don't have fixed size and it can be changed dynamically,so they can grow or shrink as required.

- Vectors can resize itself automatically when an element is inserted or deleted depending on the need of the task to be executed. It is not same in an array where only a given number of values can be stored under a single variable name.

- Vectors are assigned memory in blocks of contiguous locations. When the memory allocated for the vector falls short of storing new elements, a new memory block is allocated to vector and all elements are copied from the old location to the new location.

- This reallocation of elements helps vectors to grow when required. However, it is a costly operation and time complexity is involved in this step is linear.

Example to vector idea "as an implementation": https://onlinegdb.com/S1-YW8tqL

Modifiers

- assign : Assign vector content
- push_back : Add element at the end
- pop_back : Delete last element
- insert : Insert elements
- erase : Erase elements
- swap : Swap content
- clear : Clear content

https://onlinegdb.com/SyUC2osqU

Capacity

- size : Return size
- max_size : Return maximum size
- resize : Change size .. resize to bigger size, extra elements would be 0 by default can changed by using resize(n , fill_with_this_value)
- capacity: Return size of allocated storage capacity
- empty : Test whether vector is empty

Element access:

- operator[] : Access element
- at : Access element
- front : Access first element
- back : Access last element
- data : Access data, return a pointer to the first element in the vector

Capacity:
https://onlinegdb.com/BJdwZtMjL
Element access:
https://onlinegdb.com/r1XY7ItqL

8. Input/output with files

- ofstream: Stream class to write on files
- ifstream: Stream class to read from files
- fstream: Stream class to both read and write from/to files

```cpp
#include<iostream>
#include<fstream>

using namespace std;

int main()
{

    ofstream myfile;
    myfile.open ("first_ex.txt");    //or ofstream myfile("first_ex.txt");
    myfile << "Its C/C++ session";
    myfile.close();

    return 0;
}
```

write: https://onlinegdb.com/HygvfMucl

read: https://onlinegdb.com/H1_07fOcl