Learning the concept of inheritance, it was possible to use the member variables and functions of the parent class, depending on which types use. The keywords that I can decide when I receive inheritance are public, private, and protected. The class that gives inheritance is called the parent class, or the default class. Inherited classes are called child classes or derived classes. The relationship between base class and derived class is called "Is - a" relationship.Derived classes contain not only their own instance variables and functions, but also variables and functions from the base class. When defining the most important derived classes, they use colons, keywords, and names of parent classes. For example, class Truck : public MotorVehicle. And there is a rule between the derived class and the parent class. When creating an instance of the derived class, the constructor of the derived class is always called after calling the constructor of the base class. However, when the instance of the derived class is terminated, the destructor of the derived class is called first, followed by the destructor of the base class. Finally, importantly, it is possible to call the constructors of the base class directly from the constructors of the derived class. For example, when I create an instance of derived class using a variable in base class that is not in derived class. At this time, I can declare like Truck(const char \*\_license, int \_year, double \_capacity, const char \*\_address) : MotorVehicle(\_license, \_year). This is similar to when the derivative class was first declared.

And finally, in this workshop, I overloaded the input and output and implemented them to fit my code. These functions were also inherited from the base class and used in the derived class. In conclusion, I thought that using inheritance during this workshop could be an advantage. I believe that I can connect base class and other derived classes that plays a similar role with base class and avoid unnecessary variable declarations and function declarations. Of course, it is hard to make a complicated structure because my skills are not good enough yet. Even so, I was able to clearly understand the concept of inheritance during the workshop, and I wanted to practice it in more diverse situations and make it definitely mine.

In addition, And as I progressed with milestone 1, I could get a value from the user and use that value to get used to implementing the desired task. Also, I could practice both of the cases of receiving a value using string or using char. I realized the getcstr function without using string, but when I made it using string, I found it more convenient and fun. Personally, I think string is one of the advantages of c++. And I am not embarrassed about how to handle buffer because now I have known what is the buffer and what is in there.

Learning the concept of inheritance, I was able to use the member variables and functions of the parent class depending on which types use. The class that gives inheritance is the parent class. And the child class is the inherited class. We can call and use the functions from the parent class, but the parent cannot use from child class. To add on, when the instance of the derived class is terminated, the destructor of the derived class is called first, followed by the destructor of the base class. Also, I was able to overload the input and output and implemented them. Some trouble that I had while completing this workshop was in main line 87. Cargo of 3456 would put truck over capacity of 5432, if cargo is bulk then can add up to 5432 but if cargo is not bulk it cannot add. The sample output seems to treat it as bulk. Furthermore, the code will not allow a partial unload.