I was able to learn how to copy an object during this workshop. First, there are two ways to copy an object. There is a way to copy using a constructor or by using the operator "=". In the process of copying, shallow and deep copies occur. However, the disadvantage of shallow copying is that if a variable with dynamic memory is copied, a memory leak will occur later. This is because when copying, only pointers are copied, not values. So I used deep copy in this workshop. First, the copy constructor receives a constant-type object reference as a parameter such as Stats(const Stats &src). The values of the references received are then copied to the "this" object using a deep copy. The important point of copying is to check "self-assignment". In other words, the copy was carried out only if an object does not copy itself. The method using the operator "=" also receives a constant-type object reference as a parameter, such as a copy constructor. For example, Stats &operator=(const Stats &src). The difference between them is that the operator has an object reference as a return type. And in this workshop, I overwrote the value of the "this" object with the value of the operand object, unlike the copy constructor. In other words, I deleted the dynamic memory of the "this" object, reassigned the size of dynamic memory in the operand object, and copied it deeply. Of course, I checked "self-assignment" here as well, too. Lastly, I returned "this" object.

I have learned how to read and write files. In this workshop, I did not just read and write files, but overloaded insertions and extraction operators. At this time, it received both ostream and istream as parameters and received object references. The two forms were used as "helper functions" such as std::ostream & operator <<(std:ostream & ostr, const Stats &src) and std::istream & operator>>(std:istream &istr, Stats &src). These operators were passed as a parameter to a function defined within the class of reading and writing the file and used to enter or read the values of the member variables. And it is better to make a habit of checking with is\_open() to make sure that you read the file properly before reading the file. When I read the value in the file using getline(), I read it using string because the length is unknown. I need to convert the string to C-style using "c\_str" because to enter the value to the character pointer variable later. To use these string functions, I must declare the string library.

I was able to learn how to copy an object during this workshop. First, there are two ways to copy an object. There is a way to copy using a constructor or by using the operator "=". In the process of copying, shallow and deep copies occur. However, the disadvantage of shallow copying is that if a variable with dynamic memory is copied, a memory leak will occur later. This is because when copying, only pointers are copied, not values. So I used deep copy in this workshop. I have learned how to read and write files. In this workshop, I did not just read and write files, but overloaded insertions and extraction operators. These operators were passed as a parameter to a function defined within the class of reading and writing the file and used to enter or read the values of the member variables. And it is better to make a habit of checking with is\_open() to make sure that you read the file properly before reading the file. When I read the value in the file using getline(), I read it using string because the length is unknown. I need to convert the string to C-style using "c\_str" because to enter the value to the character pointer variable later. To use these string functions, I must declare the string library.