This lecture explains the eighth homework assignment.

For this assignment, you need to write a program that can read and write arrays of objects using F read and F write.

F read can read a large chunk of data from a file to memory. It has four arguments. The first argument is the starting address of a piece of memory. This memory must be allocated in advance and must be large enough. F read can read data from a file and store the data into an array. The second argument is the size of each element. The third argument is the number of elements. The fourth argument is a file pointer.

F write does the opposite. It takes the data from a piece of memory and saves the data into a file. The first argument is the starting address of this piece of memory. The second argument is the size of each element. The third argument is the number of elements. The fourth argument is a file pointer.

The two functions return how many elements have been successfully read or written. In an ideal scenario, the returned value should be the same as the third argument of the function. However, the returned value may be different. For example, if a file does not contain enough data to read, then returned value can be smaller. For F write, if the disk is full, then some data cannot be written to the file successfully.

How do F read and F write actually work? This figure illustrates the concept. F read takes data from a file and stores the data in memory. F write takes data from memory and stores the data in a file. The amount of data is specified by the two arguments: the size of each element and the number of elements.

Please notice that it is possible to read from the middle of the file.

Let’s see how to use F read in this example. Consider the vector structure we have seen before. Before using F read, we have to make sure enough memory is available. The program opens a file. Obviously, you need to check whether F open is successful or not. Here, to save space, I am not showing the checking code.

This is an example of using F read. The first argument is the starting address of memory. The second argument is the size of the structure. The third argument is the number of elements. The last argument is the file pointer.

After F read, check whether the number of elements actually read is the same as the third argument. If they are different, you need to handle this situation.

The next question is whether the file has any data left after F read. Sometimes additional data means problems and your programs need to handle these problems.

After everything from the file is done, please remember to F close the file.

What are the advantages using F read or F write? Let’s compare them with what we have learned before. We learned F get C, . F scan F. . F get S. . F print F ..

F read and F write can read or write large amount of data at once. By specifying the values in the second and the third arguments, it is possible reading or writing as many bytes as you wish.

One important advantage is that F read and F write can handle data of different types, as long as the types are defined inside a structure. This makes F read and F write very useful. An even more important advantage is that F read and F write can automatically handle the situation if the structure is changed. If you add or remove attributes inside the structure, you do not need to modify the program. The program still works, as long as you recompile the program. G. C. C. will calculate the size of the structure for you.

It is important using F read and F write together.

You should not use F print F to write to a file and then use F read later.

You should not use F write to write to a file and then use F get C, F scan F, or F get S. later.

If you mix them, it is very likely to make mistakes.

Let’s go back to answer an important question: how do we know how much data to read? One solution is to use F seek to go to the end of a file, and use F tell to find out the size of the file.

After opening a file, use F seek with SEEK END as the second argument and 0 as the third argument. This moves to the end of the file. At this location, F tell returns the number of bytes from the beginning of the file. If you take this value and divide it by the size of a structure, then you can get the number of element in the file.