More on structs, and Introducing file I/O

File I/O:

Why it's useful:

- practically everything on a computer is in a file, so being able to write programs that manipulate files radically enhances what we can now do.
- We can now do stuff that persists after the end of the run of the program.
- We can access stuff in the directories we have read-permission for.

File I/O in C++

```
#include <ifstream>   // input-only file functions
#include <ofstream>   // output-only
#include <fstream>    // input and output
```

File I/O in C++

```
#include <fstream>  // input and output
using namespace std;

int main ()
{
    ofstream myfile ;   // myfile is identifier for the
                        //                       file
    myfile.open ("welcomeMsg.txt");
    myfile << "Welcome to Gara's game!";
    myfile.close ();
    return 0;
}
```

The file identifier is the name it has in the program, whereas this is the name it has "outside" in the file system. It can be a path.

eg: `myfile.open ("msgs/welcomeMsg");`

File I/O in C++

```
#include <fstream>   // input and output
#include <iostream>
using namespace std;

int main ()
{   string stmt;
    ofstream logfile ;   // myfile is identifier for the file
    logfile.open ( "log.dat", ios::app) ;
    cout << "What statement do you want to log?\n";
    getline ( cin, stmt);
    logfile << stmt << endl ;
    logfile.close () ;
    return 0;
}
```

The options are :

ios:: in	- for input operations
ios:: out	- for output operations
ios:: binary	- open in binary mode
ios:: ate	- set initial position at the end
ios:: app	- all output will append
ios:: trunc	- if already exists and opened for output, deletes existing content

myfile.is_open() can be used to test if a in-file or out-file is open.

binary is used for writing blocks of memory, not necessarily text-based.

$\boxed{\text{structs}}$ … and generally getting and putting into complex data structures.

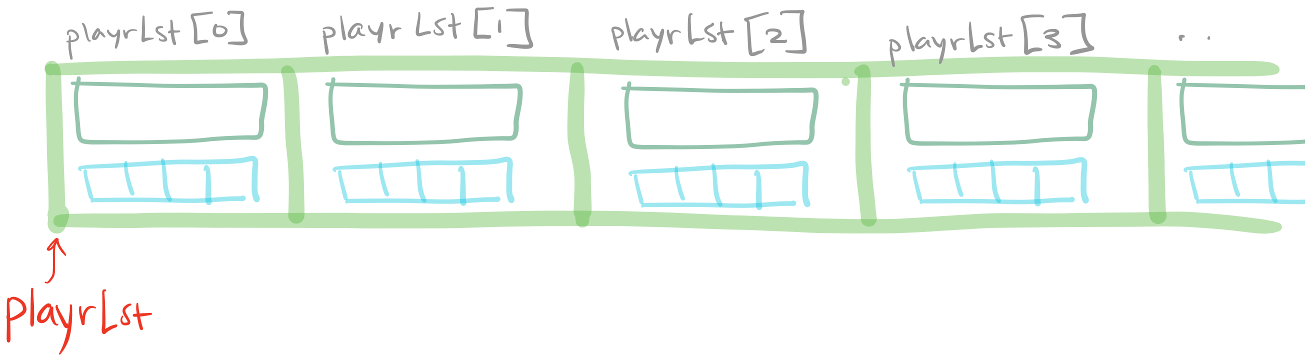
An entire item of type myStruct can be copied to another item of type myStruct, but otherwise …

$$\left.\begin{matrix} = \\ >> \\ << \end{matrix}\right\}$$

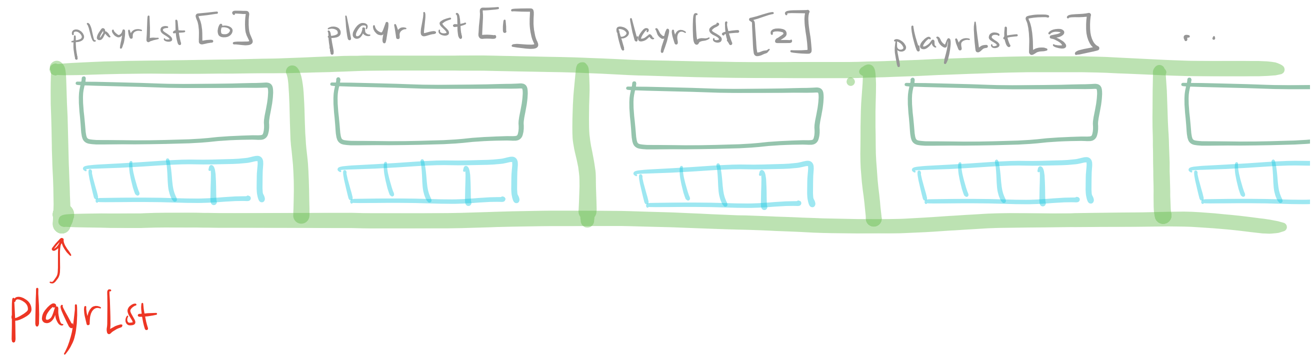
to "get" or "put" values to structs or array elements must be done

element-by-element (arrays)

field-by-field (struct)



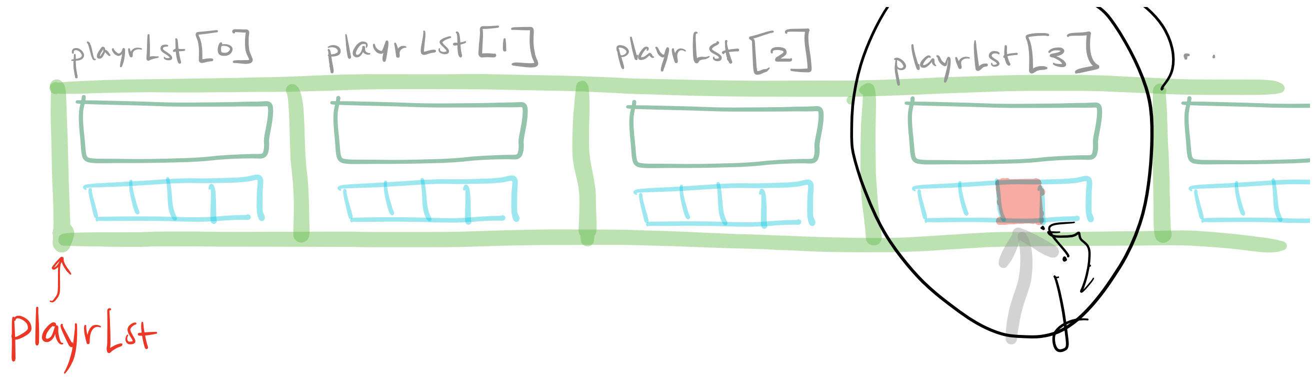
Player playrLst[maxPlayers]



Player playrLst [maxPlayers]

To write into each players struct, we use a while loop One iteration per player

and for a given player, we use a for loop to record their score One iteration per score



Player playrLst [maxPlayers]

To write into each players struct, we use a while loop One iteration per player using loop index i

and for a given player, we use a for loop to record their score one iteration per score use loop index j