**Static variable & function**

Hi, a simple explanation of static members is the keyword: “**Global”**

You can think this way, you create a variable in a class which can be shared by everyone just like a global variable, the “everyone” any objects or functions even different type of object or function. Let’s see the example.

#include <iostream>

#include <string>

using namespace std;

class Lakers\_player{

public:

**static int team\_score;**

void two\_point\_shot(){

team\_score += 2;

}

void three\_point\_shot(){

team\_score += 3;

}

int total\_score()

{

return team\_score;

}

static void print\_score()

{

cout << "team score is: " << team\_score << endl;

}

};

**int Lakers\_player::team\_score = 0;**

I define a class “Lakers\_Player” which I want to declare many players in the main function like this:

int main(){

Lakers\_player p1, p2, p3;

}

Notice I declare a static variable “team\_score” which I want to keep recording the total score of all Lakers players. And two functions to add 2 or 3 points to the “team\_score”

Important:

Once you define a static variable, you must need to do initialize for it and you have to do it outside(after) of the class, similar like you have a prototype in the class and real definition outside the class, but you need to mention which class the static variable belong to like this:

**int Lakers\_player::team\_score = 0;**

Ok, now you are ready to use it, let’s declare some Lakers\_player in the main and let the player’s add some scores.

int main(int argc, const char \* argv[]) {

Lakers\_player p1, p2, p3;

p1.two\_point\_shot();

p2.three\_point\_shot();

p3.two\_point\_shot();

cout << p1.total\_score() << endl; //output : 7

cout << p2.total\_score() << endl; //output : 7

p3.print\_score(); //output : team score is: 7

cout << Lakers\_player::team\_score << endl; //output is 7

return 0;

}

Since the player p1 and p3 both add 2 points and p2 add 3 points to the team\_score, then we can print out the static variable by just directly access it

(it’s a public member of the class) or you can call the function print\_score which we define the the class.

Another way to access the static variable just like how you call a global variable, the only different is you need to mention which class the variable belong to:

cout << **Lakers\_player::team\_score** << endl;

We will discuss more in SI sessions include static functions. Very similar concept and just let me know if you have any questions.