**It’s just a reminder of you about array initialization.**

The problem happens when people create an array without initializing cause invisible problem in their project.

When you declare an array, the system will find a sequence of memory space reserve for the array. Sometimes there are already some garbage value existing in the memory address of your array. If you directly using the array for some control statement, it might cause problem.

**Example:**

In your lab project last week, you create a string array for parking lot. Then you have a function to check if the parking lot array is full or empty. If you didn’t clean (initialize) your array in the beginning, the isFull function may give you the wrong result. Because the compiler consider the garbage is also data in the array. So although there are some spaced haven’t been used. It will still return true (means the parking lot is full). The easiest way to avoid the problem is initializing the array when you declare it.

**Here is how you can do:**

Initial the whole array as empty or all zero:

Example: int[10] = {}; // The array be set to all zero {0,0,0,0,0,0,0,0,0,0}

string park[5] = {}; //The array be set to all empty {“” , “” , “” , “” , “”}

Initial the whole array with different content:

//directly assign value for char type array, each value should be closed by ‘ ‘

char cc[6] = { 'A', 'B', 'C', 'D', 'E', 'F' };

//directly assign value for char type array, each value should be closed by “ ”

string name[3] = {“Michael”, “Judy”, “Steve”};

//directly assign value for int array

int age[6] = {33, 51, 14, 19, 29, 40};

There are some other type of arrays, sometimes you need to use a for loop to assign value in the array. Example: struct array