SBRM aka RAII:

Scope Based Resource Management:

Resource Acquisition Is Initialization

We don't need to worry about underlying resource manager

we only need to focus on higher level logic

eg: Modern C++ can ① close the file we opened before ② unlock the mutex when we leave the scope. ③ delete dynamically allocated array

this is done by the de constructor of corresponding classes (which is the most different C++ from C)

Based on this, std::string is an encapsulation of C-style char always iterate (which we have to deal with with the size management. (Memory overhead is a drawback)

(2): char sequence Std::string.

char myword2[] = { 'H', 'e', 'l', 'l', 'o', '\n' }

equivalent to.

char myword[] = "Hello";

Notice if we use string ("xxx") we don't have to take care the last '\n'

char seqs cannot be reassigned after declared:

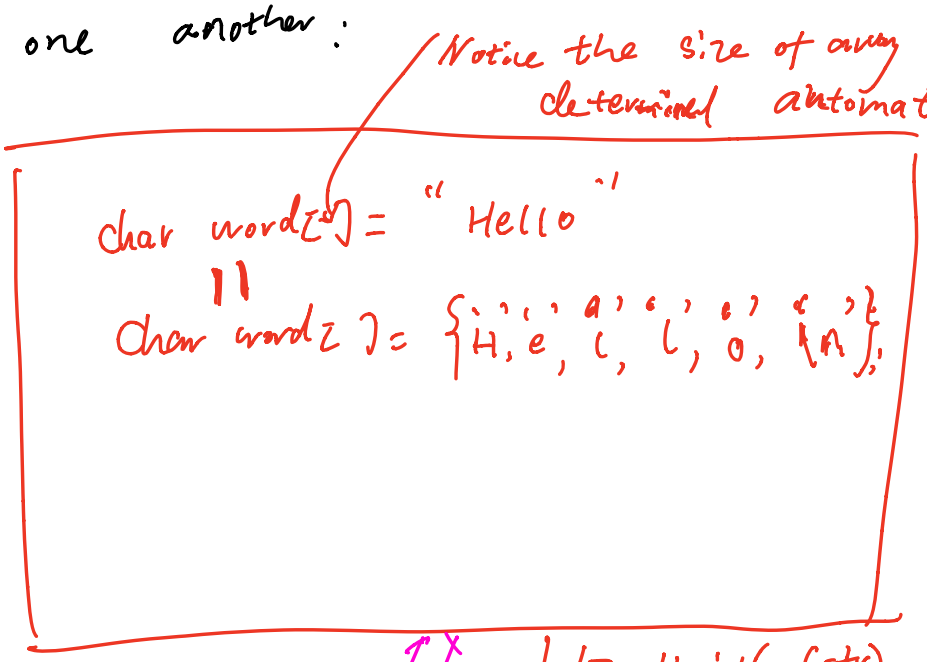
✗ myword = "Bye"; ✗

✗ myword[] = "Bye"; ✗

But elements can be reassigned individually.

myword[0] = 'B';

Conclusion: Null-terminated sequences of chars and string are interchangable. they can be transformed from one another:



eg:

```
char word[] = "Hello";
string str = word; // convert C-style to string
cout << str << endl; // Print as a library string.
                     // Print as C-string.
```

const char* str = "Hello";

③ char *. is a pointer to a constant string literal.

const char* word = "Hello";

We can change the pointer to point to something

else but cannot change value present at str.

(const) char* str = "Hello";

str[0] = 'O'; // Runtime error

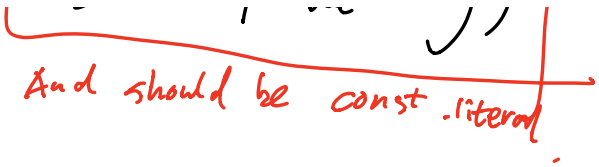
④ Differences;

In the stack section

we can modify elements in char sequence

(char word[]) but we can't do that

in (char*) ——— (In the read-only code section of the memory)


And should be const literal