Section 6: Strings (`std::string`)

Goal: create strings, use common member functions, and understand indexing rules & pitfalls.

Constructing `std::string`

```
// Strings live in the standard library
#include <iostream>
#include <string>
using namespace std;

int main() {
  string s1 = "Hello world";       // copy from string liter
  string s2("Another string");     // direct-initialize
  string s3(8, '!');               // "!!!!!!!!"
  string s0;                       // default-constructed: e

  cout << s1 << "\n" << s2 << "\n" << s3
       << "\n" << s0 << endl;
}
```

Member functions (first pass)

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
  string s = "Hello world";
  cout << s.size()   << "\n";  // number of chars
  cout << s.length() << "\n";  // same as size()
  cout << s.empty()  << "\n";  // true if length==0
  cout << s.at(0)    << "\n";  // bounds-checked access
  cout << s.substr(0, 5) << "\n"; // "Hello"
}
```

Pattern: `<instance>.<member>(args)`

Indexing with [ ] and at( )

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
  string alphabet = "abcdefghijklmnopqrstuvwxyz";
  cout << alphabet[0]      // 'a'
       << alphabet[1]      // 'b'
       << alphabet[2]      // 'c'
       << alphabet[3];     // 'd'

  // last character: use length()-1 (0-based indexing)
  cout << alphabet[alphabet.length() - 1] << endl;
}
```

Indexing with [ ] and at( )

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
  string alphabet = "abcdefghijklmnopqrstuvwxyz";
  cout << alphabet[0]     // 'a'
       << alphabet[1]     // 'b'
       << alphabet[2]     // 'c'
       << alphabet[3];    // 'd'

  // last character: use length()-1 (0-based indexing)
  cout << alphabet[alphabet.length() - 1] << endl;
}
```

Indexing is 0-based → last index is `length() - 1`.

Mutating characters

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
  string s = "abcdefghijklmnopqrstuvwxyz";

  char fifth = s[5];      // 'f'
  s[8]  = 'I';            // change 'i' -> 'I'
  s[12] += 'A' - 'a';     // make 'm' uppercase via offset
  s.at(16) = 'Q';         // bounds-checked change
  s.at(20) += 'A' - 'a';  // 'u' -> 'U'

  cout << fifth << ' ' << s << endl;
}
```

Out-of-bounds rules (important!)

- `s[pos]` with `pos >= s.length()` → **undefined behavior**.
- `s.at(pos)` with `pos >= s.length()` → **runtime error** (throws / terminates).
- Writing `s[s.length()] = ch` on a non-const `s` and non-`'\0'` `ch` → **undefined behavior**.

```
// Danger zone demo — don't run in production!
string s = "0123";
// s[28];          // undefined behavior
// s.at(28);       // runtime error
```

Prefer `at()` while learning; switch to `[]` when you are sure indices are valid.

`size()/length()` return type

- Return `std::size_t`, an **unsigned** integer type.
- Large enough to index any element of the string.

```cpp
#include <iostream>
#include <string>
#include <cstddef>    // for std::size_t
using namespace std;

int main() {
  string s = "hello";
  std::size_t       n1 = s.length();
  std::size_t       n2 = s.size();
  size_t            i1 = 0;

  cout << n1 << ' ' << n2 << ' ' << i1 << endl;
  cout << s[i1++] << ' ';
  cout << s[i1++] << "\t ";
  cout << s[i1++] << endl;
```

find (also size_t)

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
  string s = "bananas";
  size_t pos = s.find("ana");
  std::cout << pos << '\n'; // 1
}
```

Section 7: The Input Buffer

Understand how `std::cin` consumes characters and how to combine >>, `getline`, and helpers safely.

Example

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    string first_name, last_name;
    cout << "First name? ";
    cin >> first_name;          // reads up to whitespace
    cout << "Last name? ";
    cin >> last_name;           // reads up to whitespace
    cout << "Hello, " << first_name << " " << last_name <<
}
```

Types `Liyao`, press `Enter`, types `Lyu`, press `Enter`.

Example

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    string first_name, last_name;
    cout << "First name and last name? ";
    cin >> first_name >> last_name;  // reads up to whitesp
    cout << "Hello, " << first_name << " " << last_name << 
}
```

Types `Liyao  Lyu`, press `Enter`.

Example

```cpp
#include <iostream>
#include <string>
using namespace std;
int main() {
    string full_name;
    cout << "Full name? ";
    getline(cin, full_name);          // reads entire line
    cout << "Hello, " << full_name << "!\n";
}
```

Types `Liyao UCLA CA ... Lyu`, press `Enter`.

Failed Example

```cpp
#include <iostream>
#include <string>
using namespace std;
int main() {
    cout << "Favorite number? ";
    int favorite_number; cin >> favorite_number;
    cout << "Full name? ";
    string full_name; getline(cin, full_name);
    cout << "Hello, " << full_name << "! Your favorite numb
}
```

Fixed Example (fixed)

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    cout << "Favorite number? ";
    int favorite_number; cin >> favorite_number;
    cin.ignore();                // discard rest of line
    cout << "Full name? ";
    string full_name; getline(cin, full_name);
    cout << "Hello, " << full_name << "! Your favorite numb
}
```

Input Buffer

- Programs start with an empty input buffer.
- After users have entered input and pressed `Enter`, a program can make use of the typed charcters via the input buffer
- A program may or may not use all of the characters in the input buffer by the time it finishes executing.

Example

```cpp
#include <iostream>
using namespace std;
int main() {
    cout << "Favorite number? ";
    int favorite_number; cin >> favorite_number;
    cout << "Your favorite number is " << favorite_number <<
    return 0;
}
```

Definition

Here are the instructions that `cin >> variable;` performs:

1. If there are no characters in the input buffer, the program waits for the user to enter a value.
2. Remove leading whitespace characters from the input buffer. (different from textbook,explain latter)
3. Repeat until a non-whitespace character is found.
4. **Starting at the beginning of the input buffer, interpret as many characters as possible as the desired type** (e.g., int, double, string).

Example

When `variable` is type `int`, and the input buffer contains `"90024UCLA\n"`:
The characters "90024" are extracted and converted to the integer value 90024. The remaining characters "UCLA\n" are left in the input buffer.

Example

When `variable` is type `int`, and the input buffer contains `"900.24UCLA\n"`:
The characters "900" are extracted and converted to the integer value 900. The remaining characters ".24UCLA\n" are left in the input buffer.

Example

When `variable` is type `double`, and the input buffer contains `"900.24UCLA\n"`:
The characters "900.24" are extracted and converted to the double value 900.24. The remaining characters `"UCLA\n"` are left in the input buffer.

Example

When `variable` is type `char`, and the input buffer contains `"900.24UCLA\n"`:
The characters "9" are extracted and stored in the char variable. The remaining characters "00.24UCLA\n" are left in the input buffer.

Example

When `variable` is type `string`, and the input buffer contains `"900.24U CLA\n"`:
The characters "900.24U" are extracted and stored in the string variable. The remaining characters " CLA\n" are left in the input buffer.

Definition

Suppose that `string` called `str` has been constucted.
Here are the instructions that `getline(cin, str);` performs:

1. If there are no characters in the input buffer, the program allow the user to enter a value.
2. Extract characters from the input buffer before `\n` (newline character) and store them in `str` . When the input buffer only contains '\n', the empty string is stored in `str` .
3. Remove the `\n` from the input buffer, but do not store it in `str` .

Definition

Here are the instructions that `cin.ignore();` performs:

1. If there are no characters in the input buffer, the program allow the user to enter a value.
2. Remove the first character from the input buffer.