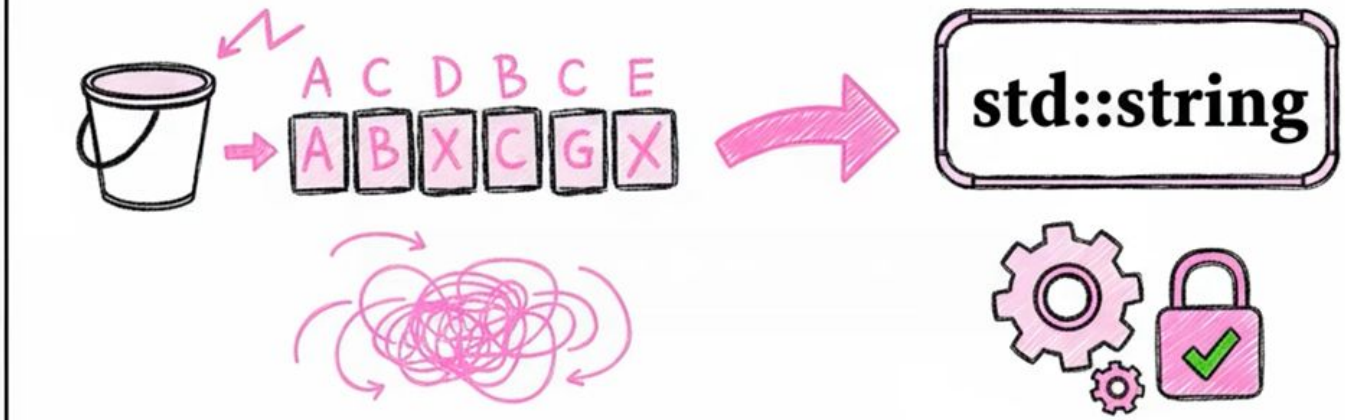
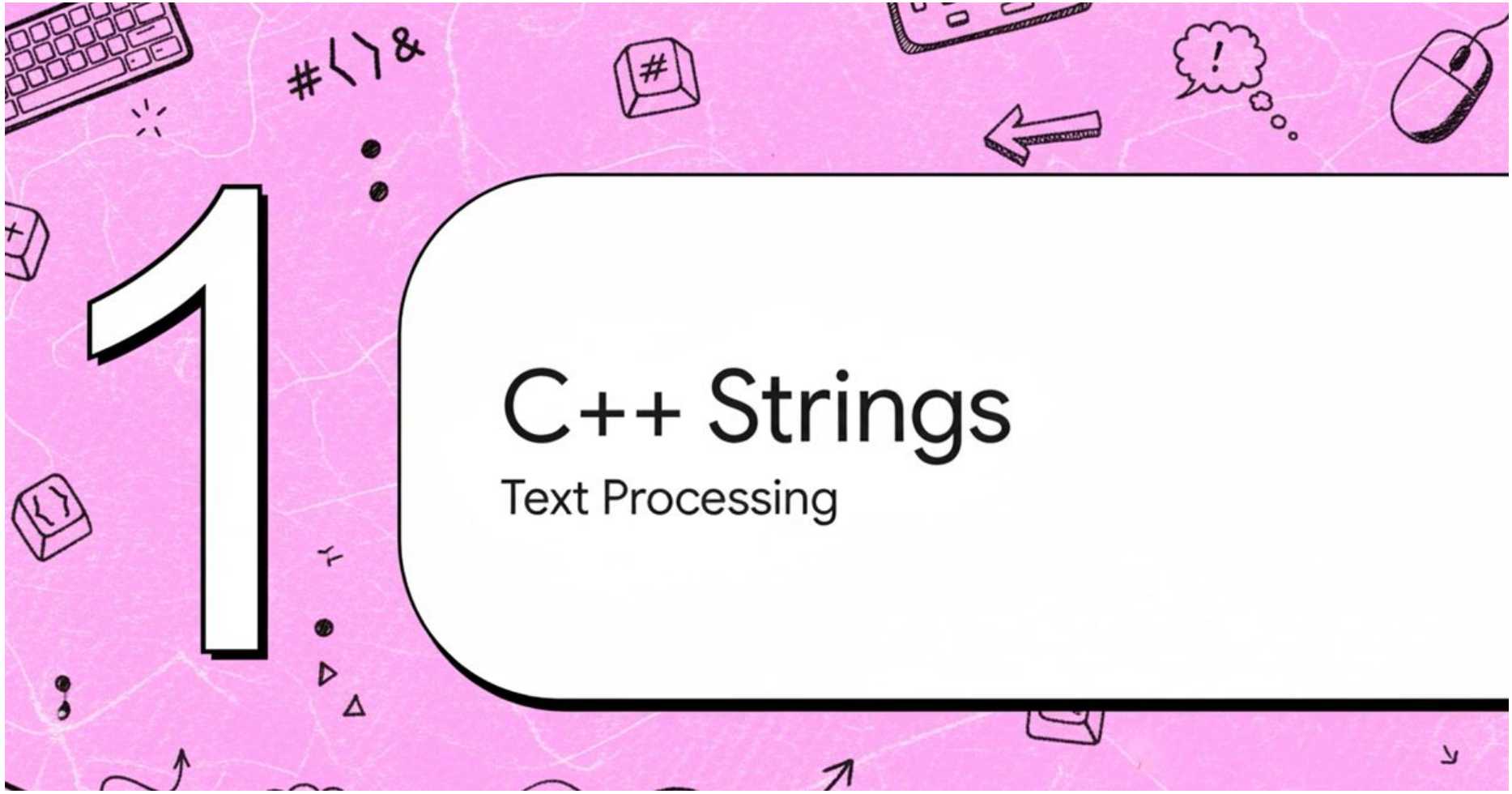
Week 10-11

Strings and Files

The Explainer




1
C++ Strings
Text Processing


How do programs
understand and
manage text?
NULL

01

What Is a String?

02

The Classic C-String

03

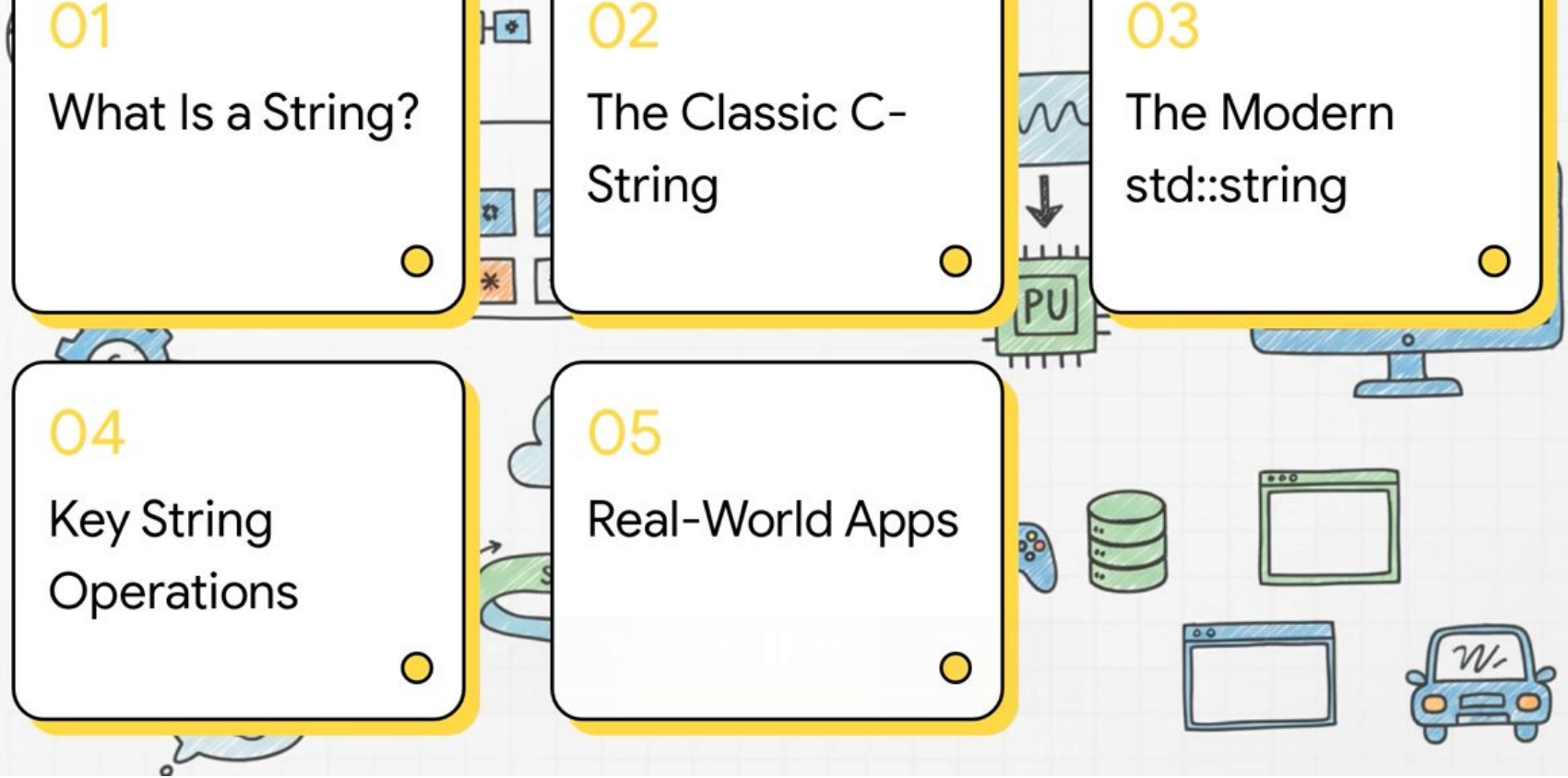
The Modern std::string

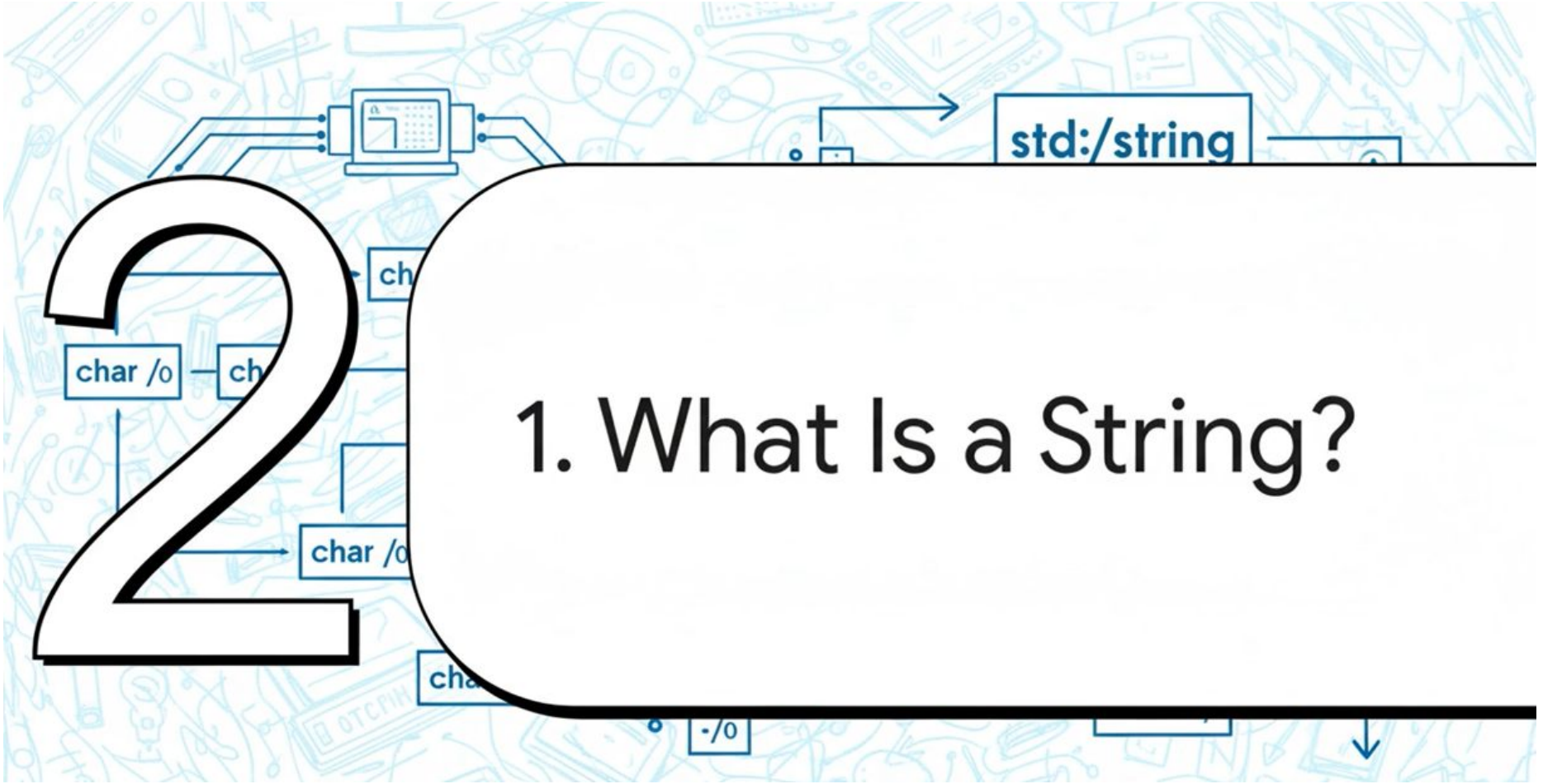
04

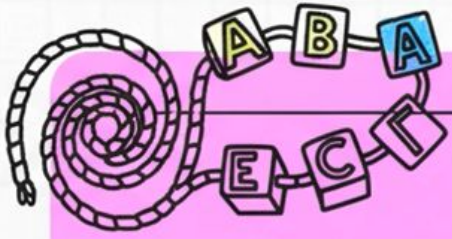
Key String Operations

05

Real-World Apps



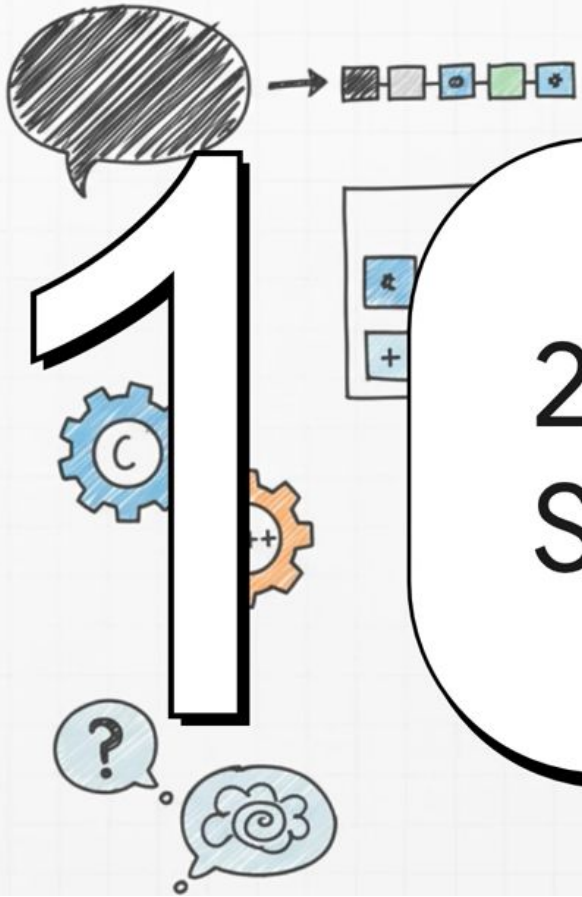

2
std:/string
char /0
char /0
1. What Is a String?



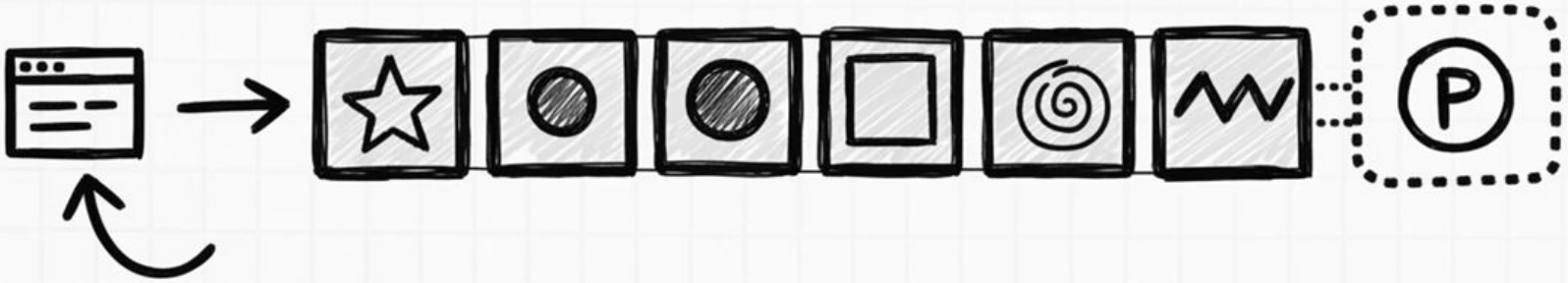
String

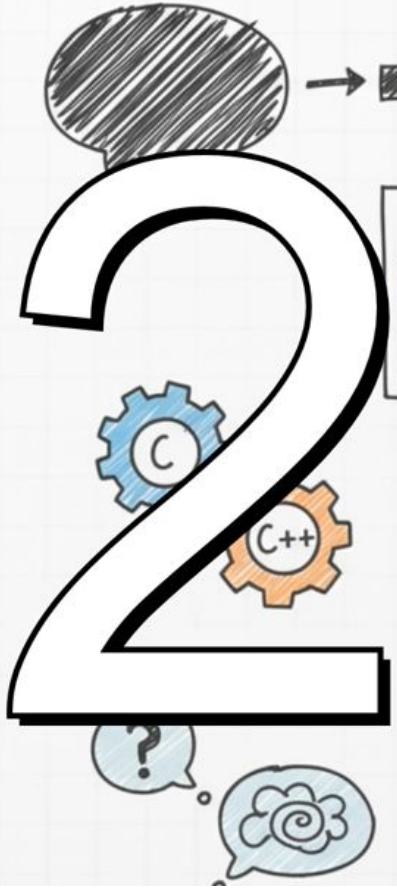
A sequence of characters. In C++, anything enclosed in double quotes is a string.

Examples of Strings

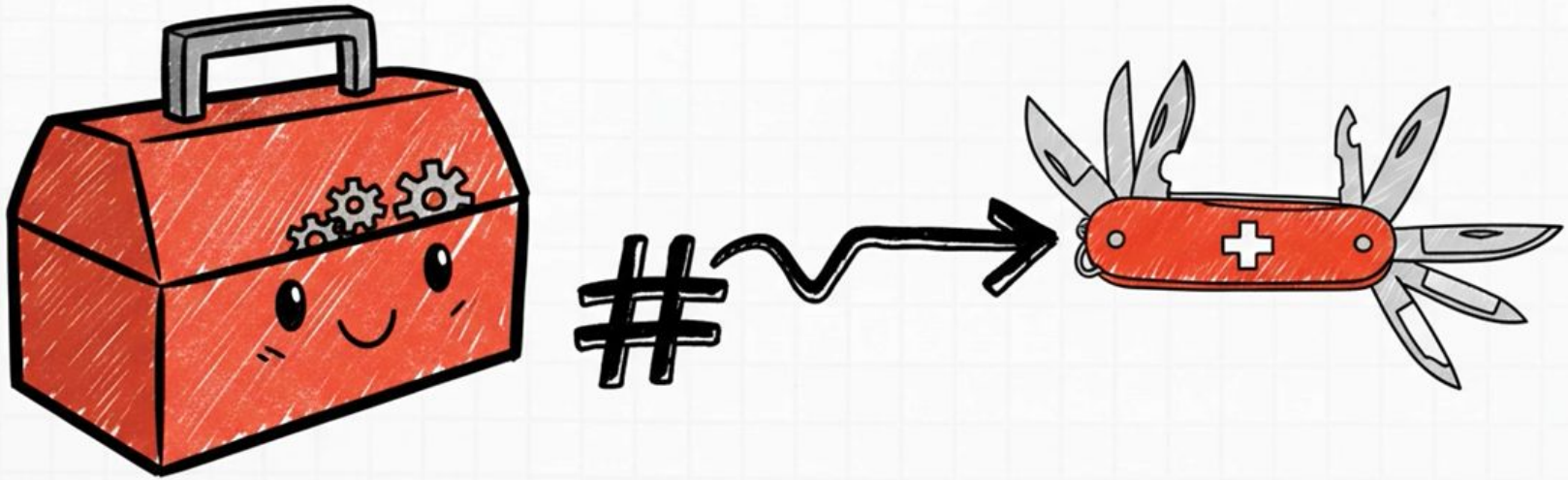


2. The Classic C-String


P



3. The Modern std::string



C-String (`char[]`): A fixed-size array inherited from C. Requires manual functions like `strlen()`.

Modern `std::string`: A flexible C++ object that manages memory and has built-in methods.

4. Key String Operations

How do you read a full line of text, including spaces?

manual
C-style
strings

Using cin.getline()

1. Declare Array

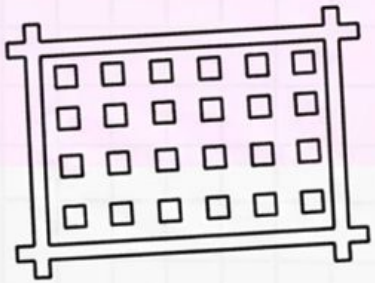
Declare a character array with a fixed size, e.g., `char text[50];`

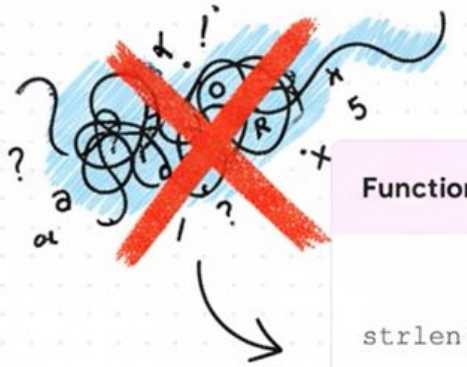
2. Clear Input

Use `cin.ignore()` to clear any leftover input from the buffer.

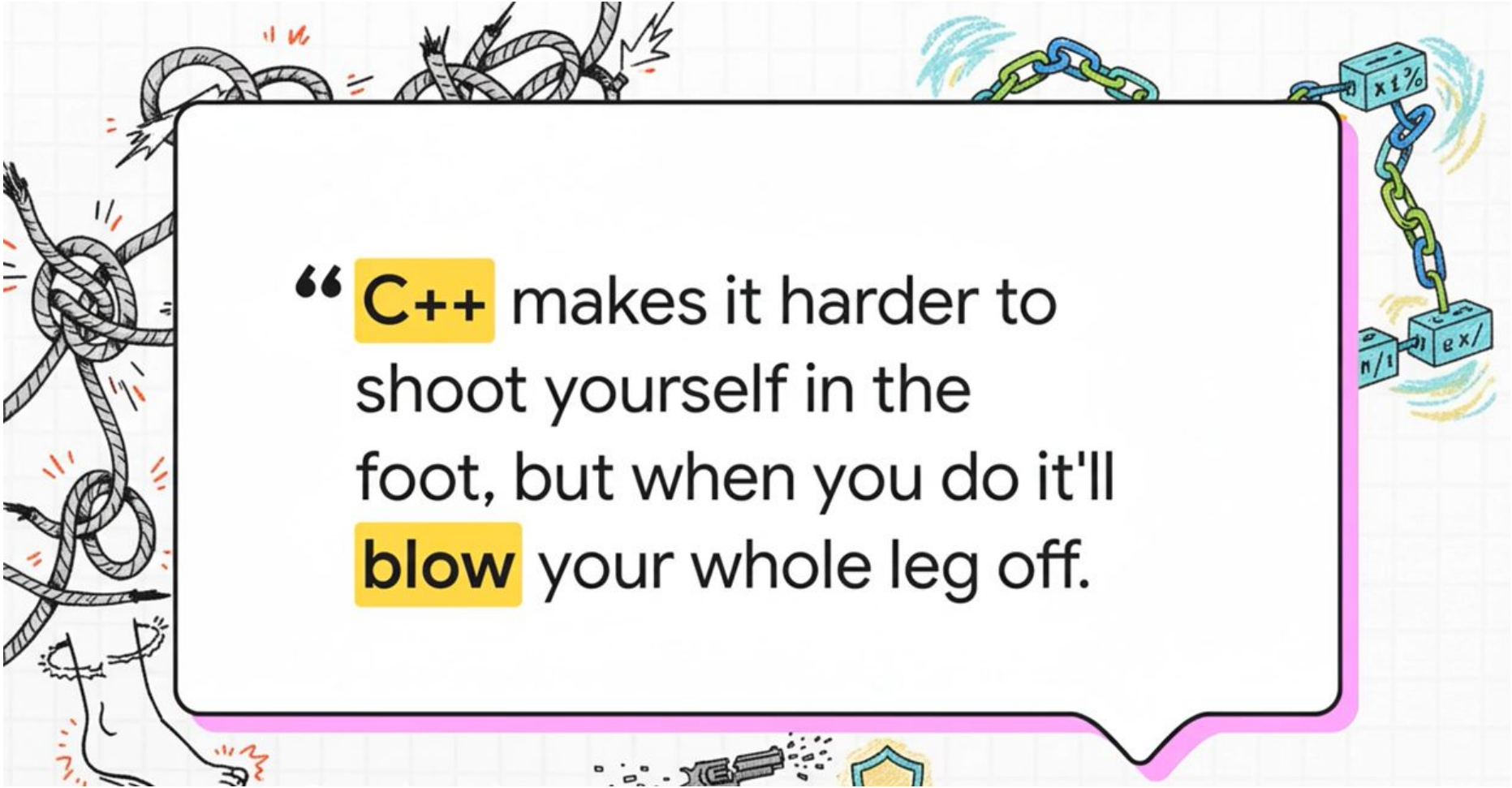
3. Call getline

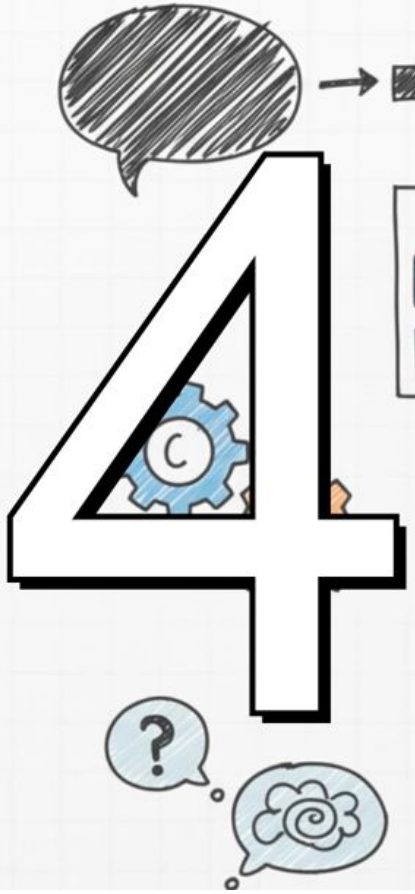
Call `cin.getline(text, 50);` to read the entire line.





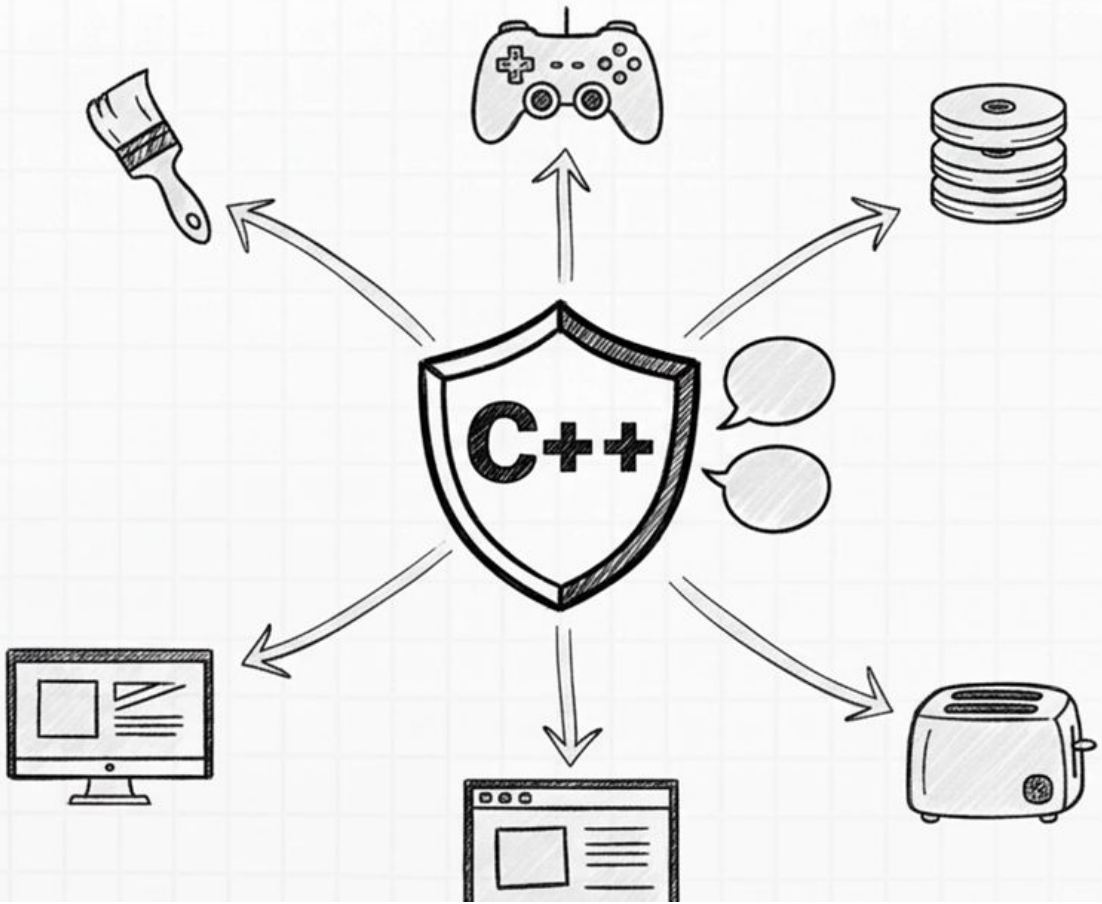
Function	Purpose	Example
`strlen(name)`	Returns the string's length	`strlen("Hello")` is 5
`strcpy_s(dest, src)`	Copies one string to another	`strcpy_s(name, "Jenny")`
`strcmp(str1, str2)`	Compares two strings	Returns 0 if they match

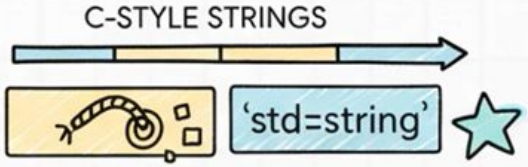

“ C++ makes it harder to shoot yourself in the foot, but when you do it'll blow your whole leg off.

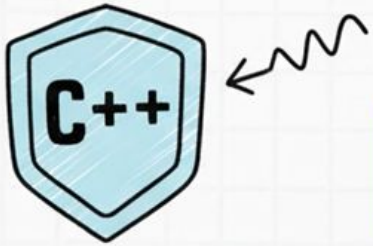


5. Real-World Apps

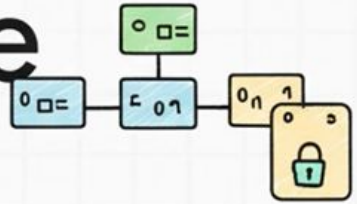
Why Strings Matter

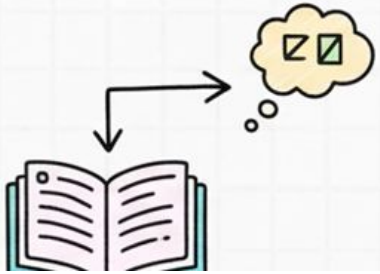

C++





Now that you can handle text, what stories will your code tell?

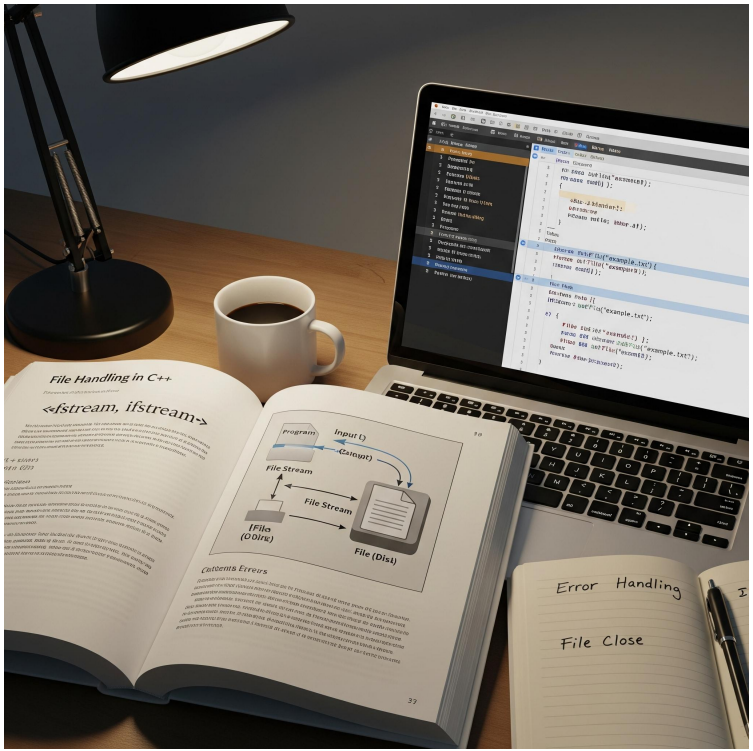




Introduction to File Processing

Understand what a file stream is.

- Learn to use <fstream> for file input/output.
- Write data to files using ofstream.
- Read data from files using ifstream.
- Handle errors and close files properly.



Writing Data to a File

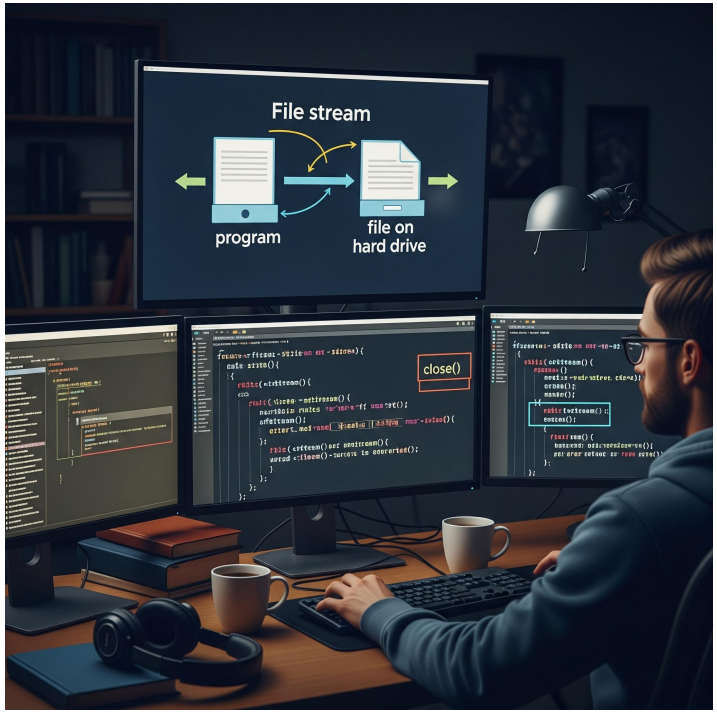
Definition: A stream is a flow of data between a program and an input/output device.

Types of Streams:

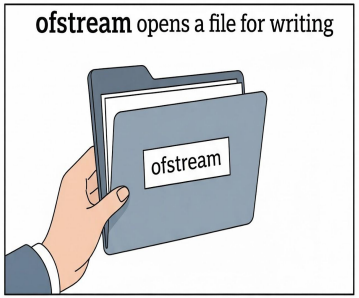
- cin – input stream from the keyboard.
- cout – output stream to the console.
- ifstream – input stream from a file.
- ofstream – output stream to a file.

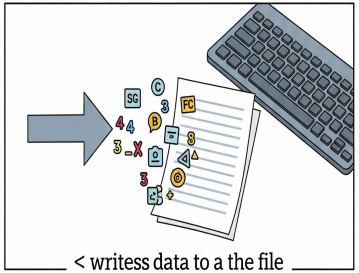
Library Required:

```
#include <fstream>
```



Writing Data to a File





Code Example:

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

int main() {
    ofstream outFile("example.txt");

    if (!outFile) {
        cout << "Error opening file." << endl;
        return 1;
    }

    outFile << "Hello, C++ file processing!" << endl;
    outFile << "Files let us save data permanently." << endl;

    outFile.close();
    cout << "Data written to example.txt" << endl;
}
```

Key Points: - `ofstream` opens a file for writing. - `<<` writes data to the file. - Always call `.close()` when done.

Reading Data from a File

Code Example:

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

int main() {
    ifstream inFile("example.txt");
    string line;

    if (!inFile) {
        cout << "File not found." << endl;
        return 1;
    }

    while (getline(inFile, line)) {
        cout << line << endl;
    }

    inFile.close();
}
```

Concepts: - `getline()` reads one line at a time. - Loop continues until the end of the file. - `.close()` ensures resources are released.

Combined File Operations

Example: Copying Data Between Files

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

int main() {
    ifstream source("input.txt");
    ofstream dest("copy.txt");

    string line;
    while (getline(source, line)) {
        dest << line << endl;
    }

    cout << "File copied successfully!" << endl;
    source.close();
    dest.close();
}
```

Concepts: - Combine reading and writing in one program. - Useful for file backups, logs, and data transformations.

Common File I/O Functions

Function	Description
`.open()`	Opens a file manually
`.close()`	Closes the file
`.fail()`	Checks for open/read/write errors
`getline()`	Reads a line from file
`<<`	Writes to a file
`>>`	Reads from a file

Error Handling in File I/O

Always check if a file opened successfully.

```
ifstream file("data.txt");
if (!file) {
    cout << "Error: File could not be opened." << endl;
    return 1;
}
```

Other Good Practices:

- Close every file you open.

- Avoid overwriting existing files accidentally.

- Check for end-of-file (eof()) conditions.

Real-World Applications

Where File I/O is Used:

- Saving user data (like game progress or settings).

- Reading datasets for analysis.

- Logging program activities.

- Data backups and reports.

Example: A student record system that saves names and scores in a .txt file.

Transition to Lab File Read

Lab File Read / Write Objective:

- Practice reading and writing data files.

- Apply loops, arrays, and conditionals to handle file data.

Lab File Read Task:

Write a program that:

1. Takes 5 student names and scores.

2. Writes them to a file students.txt.

3. Reads the file back and displays the data.

Attendance