File Handling in C++

This document will guide you through **file handling in C++**, from the basics to advanced concepts like random access and binary files. It includes examples, best practices, and exercises.

---

1. Introduction

File handling allows programs to **store, retrieve, and manipulate data** from files. C++ provides `fstream`, `ifstream`, and `ofstream` classes to work with files.

Header files required:

```cpp
#include <fstream>
#include <iostream>
#include <string>
#include <filesystem> // optional (C++17+)
```

File Stream Classes

- `std::ifstream` — input file stream (read-only)
- `std::ofstream` — output file stream (write-only)
- `std::fstream` — both input and output

---

2. Opening and Closing Files

Open for reading:

```cpp
std::ifstream in("input.txt");
if (!in.is_open()) {
    std::cerr << "Failed to open file";
}
```

Open for writing:

```cpp
std::ofstream out("output.txt", std::ios::out | std::ios::trunc);
out << "Hello World";
```

Modes of Opening

- `std::ios::in` — read
- `std::ios::out` — write
- `std::ios::app` — append
- `std::ios::trunc` — overwrite
- `std::ios::binary` — binary mode
- `std::ios::ate` — open and seek to end

---

3. Reading from Files

Line-wise:

```
std::string line;
while (std::getline(in, line)) {
    std::cout << line << "\n";
}
```

Word-wise:

```
std::string word;
while (in >> word) {
    std::cout << word << "\n";
}
```

Character-wise:

```
char c;
while (in.get(c)) {
    std::cout << c;
}
```

---

4. Writing to Files

```
std::ofstream out("output.txt");
out << "Hello, file!" << std::endl;
out << 123 << " " << 4.56;
```

You can force writing with `std::flush`:

```
out << "Important line" << std::flush;
```

---

5. Copying a File

```
std::ifstream in("source.txt", std::ios::binary);
std::ofstream out("dest.txt", std::ios::binary);
out << in.rdbuf();
```

---

6. Random Access in Files

```
std::fstream file("data.bin", std::ios::in | std::ios::out | std::ios::binary);
file.seekg(0, std::ios::end);
std::streampos size = file.tellg();
file.seekg(0, std::ios::beg);
```

```
file.seekp(100); // move write pointer
file << "Data";
```

---

7. Binary File Handling

Writing binary data:

```
struct Header {
    uint32_t magic;
    uint16_t version;
};

Header h{0xDEADBEEF, 1};
std::ofstream out("data.bin", std::ios::binary);
out.write(reinterpret_cast<const char*>(&h), sizeof(h));
```

Reading binary data:

```
Header h;
std::ifstream in("data.bin", std::ios::binary);
in.read(reinterpret_cast<char*>(&h), sizeof(h));
```

---

8. Error Handling in File I/O

Streams set **state flags**:

- `eofbit` — end of file
- `failbit` — logical error (e.g., format mismatch)
- `badbit` — I/O error

Enable exceptions:

```
out.exceptions(std::ofstream::failbit | std::ofstream::badbit);
try {
    out.open("file.txt");
    out << "data";
} catch (const std::ios_base::failure& e) {
    std::cerr << e.what();
}
```

---

9. RAII Principle

Files close automatically when streams go out of scope:

```
{
    std::ofstream out("file.txt");
    out << "Scoped write";
} // file closed automatically
```

---

10. Checking File Existence (C++17+)

```
if (std::filesystem::exists("myfile.txt")) {
    std::cout << "File exists";
}
```

---

11. Example: Log File with Timestamps

```
#include <chrono>
#include <ctime>

void log_message(const std::string& msg) {
    std::ofstream log("app.log", std::ios::app);
    auto now = std::chrono::system_clock::now();
    std::time_t t = std::chrono::system_clock::to_time_t(now);
    log << std::ctime(&t) << ": " << msg << "\n";
}
```

---

12. Best Practices

- Use `std::getline` for full lines
- Always check `if (stream)` before using
- Use binary mode for exact byte control
- Avoid mixing formatted (`>>`) and unformatted (`read`) I/O
- Use RAII: let streams close automatically
- Flush only when necessary

---

13. Complete Example: Count Lines in File

```
#include <fstream>
#include <iostream>
#include <string>

int main() {
    std::ifstream in("input.txt");
    if (!in) return 1;

    size_t lines = 0;
    std::string temp;
    while (std::getline(in, temp)) ++lines;

    std::ofstream out("summary.txt");
    out << "Number of lines: " << lines;
    return 0;
}
```

---

14. Practice Problems

1. Write a program to read a file and count the number of words.
2. Write a program to copy a binary file (e.g., image) safely.
3. Write a program that appends user input to a log file.
4. Write a program to search for a word in a text file and print its occurrences.
5. Create a student database using binary files (store `id`, `name`, `marks`).

✅With this, you now have a full overview of **file handling in C++** with theory, examples, and practice tasks.