CS2810 OOAIA

Streams

Example Code Snippets

1. Handling Input Errors with `cin`

```
#include <iostream>
#include <limits> // for numeric_limits

int main() {
    int age;
    while (true) {
        std::cout << "Enter your age: ";
        std::cin >> age;
        if (std::cin.fail()) { // Check for input failure
            std::cin.clear(); // Clear error state

std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
// Discard bad input
            std::cout << "Invalid input. Please enter a number.\n";
        } else {

std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
// Clear buffer
            break;
        }
    }
    std::cout << "Age: " << age << "\n";
```

```
    return 0;
}

// Uses cin.fail() to detect invalid input (e.g., letters instead of
numbers).
// cin.clear() resets the error state, and ignore() removes invalid
characters from the buffer.
```

2. File Copy with Line-by-Line Processing

```
#include <iostream>
#include <fstream>
#include <cctype> // for toupper()

int main() {
    std::ifstream input("input.txt");
    std::ofstream output("output.txt");
    std::string line;

    if (!input.is_open() || !output.is_open()) {
        std::cerr << "Error opening files!\n";
        return 1;
    }

    while (std::getline(input, line)) {
        for (char& c : line) c = toupper(c);
        output << line << '\n';
    }

    input.close();
    output.close();
    return 0;
}

// Uses getline() to read lines without duplicating the last line
(avoids eof() pitfalls).
// Transforms each character to uppercase before writing to the
output file.
```

3. Skipping Delimiters with `ignore()`

```
#include <iostream>

int main() {
    int day, month, year;
    char slash;
    std::cout << "Enter date (DD/MM/YYYY): ";
    std::cin >> day;
    std::cin.ignore(1, '/'); // Skip the '/' after day
    std::cin >> month;
    std::cin.ignore(1, '/'); // Skip the '/' after month
    std::cin >> year;
    std::cout << "Date: " << day << "-" << month << "-" << year <<
"\n";
    return 0;
}

// ignore(1, '/') skips exactly one / character between day, month,
and year.
```

4. Formatted Table with `setw` and `setprecision`

```
#include <iostream>
#include <iomanip> // for setw, setprecision

int main() {
    std::cout << std::left << std::setprecision(2) << std::fixed;
    std::cout << std::setw(20) << "Product"
              << std::setw(10) << "Price"
              << std::setw(10) << "Stock" << "\n";
    std::cout << std::setw(20) << "Laptop"
              << std::setw(10) << 999.99
              << std::setw(10) << 5 << "\n";
    std::cout << std::setw(20) << "Mouse"
              << std::setw(10) << 25.5
              << std::setw(10) << 100 << "\n";
```

```
    return 0;
}

// setw(n) sets column width, fixed and setprecision(2) format
floating-point numbers.

Output:

Product             Price     Stock
Laptop              999.99    5
Mouse               25.50     100
```

5. Parsing Numbers with `peek()` and `putback()`

```
#include <iostream>

int main() {
    int num;
    std::cout << "Enter numbers (end with 'q'): ";
    while (std::cin.peek() != 'q') { // Check next character without
extracting
        if (isdigit(std::cin.peek())) {
            std::cin >> num;
            std::cout << "Read: " << num << "\n";
        } else {
            std::cin.putback(std::cin.get()); // Put non-digit back
into stream
            break;
        }
    }
    return 0;
}

// peek() checks the next character without removing it.
// putback() returns a character to the stream for future processing.
```

6. Command-Line File Copier with Error Checking

```cpp
#include <iostream>
#include <fstream>

int main(int argc, char* argv[]) {
    if (argc != 3) {
        std::cerr << "Usage: " << argv[0] << " <input> <output>\n";
        return 1;
    }

    std::ifstream input(argv[1], std::ios::binary);
    std::ofstream output(argv[2], std::ios::binary);

    if (!input) {
        std::cerr << "Failed to open input file.\n";
        return 2;
    }
    if (!output) {
        std::cerr << "Failed to create output file.\n";
        return 3;
    }

    output << input.rdbuf(); // Efficiently copy contents

    input.close();
    output.close();
    return 0;
}

// Uses argc and argv to read filenames from the command line.
// rdbuf() copies the entire file content efficiently.
```

Stream Type	Class/Header	Key Functions/Methods	Purpose/Use Case	Notes
Standard Input	istream(<iostream>)	cin >> var, cin.get(), cin.ignore(n, ch)	Read data from keyboard/stdin.	- Ignores leading whitespace by default. - Use ignore() to skip delimiters.
Standard Output	ostream(<iostream>)	cout << var, cout.put(), cout.flush()	Write data to console/stdout.	- Buffered; use endl or flush to force output. - endl adds \n + flushes.
File Input	ifstream(<fstream>)	open(), close(), getline(file, str)	Read from files.	- Avoid eof() loops; prefer while (getline(...)). - Check is_open().
File Output	ofstream(<fstream>)	open(), close(), ofs << data	Write to files.	- Use ios::binary for non-text files. - Overwrites by default.
Formatted I/O	<iomanip>	setw(n), setprecision(n), fixed, left	Control output alignment and precision.	- setw(n) sets column width. - setprecision(2) for 2 decimal places.

Error Handling	All streams	fail(), clear(), good(), bad()	Detect/correct stream errors.	- Use cin.clear()after invalid input. - Check fail() before retrying.
Stream Manipulation	All streams	peek(), putback(), ignore(n, ch)	Inspect/modify stream buffer.	- peek() checks next character without extraction. - putback() undoes extraction.

Practice Problems

1. Problem: Read a string in the format Name:Age:City (e.g., Alice:25:NewYork) and split it into three variables. Handle leading/trailing spaces.

Example:

```
Input:
John : 30 : London
Output:
Name: John, Age: 30, City: London
```

2. Problem: Write a program that reads a file line-by-line and writes a new file where each line is reversed.

Example:

```
Sample Input File (input.txt):
Hello
World
Expected Output File (output.txt):
olleH
dlroW
```

Hint: Use `ifstream` and `ofstream`.

3. Problem: Continuously prompt the user for an integer between 1-100. Handle non-integer inputs and out-of-range values.

Example:

```
Sample Interaction:
Enter a number (1-100): abc
```

```
Invalid input! Try again.
Enter a number (1-100): 150
Out of range! Try again.
Enter a number (1-100): 50
Valid input: 50
```

Hint: Use `cin.fail()` and `cin.clear()` and reset the input buffer with `cin.ignore(...)`.

4. Problem: Read a CSV file (data.csv) with headers Name,Age,Score and generate a JSON-formatted output.

Example:

```
Sample Input File:
Name,Age,Score
Alice,25,95.5
Bob,30,88.0

Expected Output:
[
  {"Name": "Alice", "Age": 25, "Score": 95.5},
  {"Name": "Bob", "Age": 30, "Score": 88.0}
]
```