## BUILT-IN MANIPULATORS

#include <iostream> #include <fstream> #include <iomanip>

int main() {

std::ofstream file("example.txt");

file << std::setw(10) << "Name" << std::setw(10) << "Age" << std::endl; file << std::setw(10) << "John" << std::setw(10) << 25 << std::endl;

file << std::setprecision(2) << std::fixed << 3.14159 << std::endl; file.close();

return 0;

} OUTPUT:-



# USER-DEFINED MANIPULATOR

#include <iostream> #include <fstream>

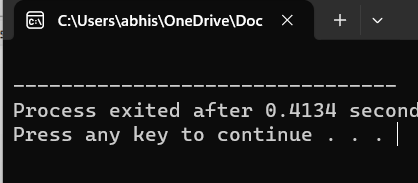
std::ostream& customManipulator(std::ostream& os) { os << "Custom Manipulator: ";

return os;} int main() {

std::ofstream file("example.txt");

file << customManipulator << "Hello, World!" << std::endl; file.close();

return 0;} OUTPUT:-



## READING NUMERICAL DATA

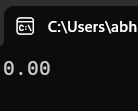
#include <iostream> #include <fstream> #include <iomanip> int main() {

std::ifstream file("example.txt"); double num;

file >> num;

std::cout << std::setprecision(2) << std::fixed << num << std::endl; file.close();

return 0;} OUTPUT:-



## BUILT-IN AND USER-DEFINED MANIPULATORS

#include <iostream> #include <fstream> #include <iomanip>

std::ostream& customManipulator(std::ostream& os) { os << "Custom Manipulator: ";

return os;

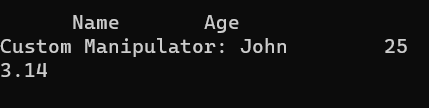
}int main() {

std::ofstream file("example.txt");

file << std::setw(10) << "Name" << std::setw(10) << "Age" << std::endl;

file << customManipulator << "John" << std::setw(10) << 25 << std::endl; file.close();

return 0;}



## FORMATTED TABLE

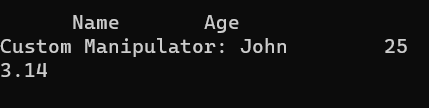
#include <iostream> #include <fstream> #include <iomanip> int main() {

std::ofstream file("example.txt");

file << std::setw(10) << "Name" << std::setw(10) << "Age" << std::setw(10) << "Grade" << std::endl;

file << std::setw(10) << "John" << std::setw(10) << 25 << std::setw(10) << "A" << std::endl; file << std::setw(10) << "Jane" << std::setw(10) << 30 << std::setw(10) << "B" << std::endl; file.close();

return 0;}



## FILE MODES

#include <iostream> #include <fstream> int main() {

std::ofstream file1("example.txt", std::ios::out); file1 << "Hello, World!" << std::endl;

file1.close();

std::ifstream file2("example.txt", std::ios::in); std::string line;

std::getline(file2, line);

std::cout << line << std::endl; file2.close();

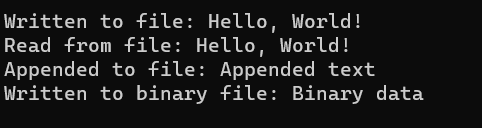
std::ofstream file3("example.txt", std::ios::app); file3 << "Appended text" << std::endl;

file3.close();

std::ofstream file4("example.bin", std::ios::binary); char buffer[] = "Binary data";

file4.write(buffer, sizeof(buffer)); file4.close();

return 0;}



## APPEND MODE

#include <iostream> #include <fstream> int main() {

std::ofstream file("example.txt", std::ios::app); file << "Appended text" << std::endl;

file.close(); return 0}



## BINARY FILE

#include <iostream> #include <fstream>

struct Student { char name[20]; int age;

};

int main() {

std::ofstream file("example.bin", std::ios::binary); Student student = {"John", 25};

file.write((char\*)&student, sizeof(Student)); file.close();

return 0;

}



## READING AND WRITING OBJECTS

#include <iostream> #include <fstream> class Student {

public:

std::string name; int age;

};

int main() {

Student student = {"John", 25};

// Write object to file

std::ofstream file("example.txt");

file << student.name << " " << student.age << std::endl; file.close();

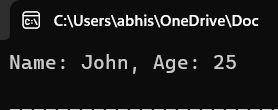
// Read object from file Student readStudent;

std::ifstream readFile("example.txt");

readFile >> readStudent.name >> readStudent.age; readFile.close();

std::cout << "Name: " << readStudent.name << ", Age: " << readStudent.age << std::endl;

return 0;}



## FILE POINTERS

#include <iostream> #include <fstream>

int main() {

std::fstream file("example.txt", std::ios::in | std::ios::out | std::ios::trunc);

// Write to file

file << "Hello, World!" << std::endl;

file << "This is a test file." << std::endl;

// Move file pointer to the beginning file.seekg(0);

// Read from file char buffer[100];

file.read(buffer, 100);

std::cout << buffer << std::endl;

// Get current file pointer position

std::cout << "Current position: " << file.tellg() << std::endl;

// Move file pointer to the end file.seekg(0, std::ios::end);

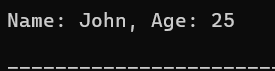
std::cout << "End position: " << file.tellg() << std::endl;

// Rewind file pointer to the beginning file.rewind();

std::cout << "Rewind position: " << file.tellg() << std::endl; file.close();

return 0;

}



## ERROR HANDLING WITH PERROR() AND FERROR()

#include <iostream> #include <fstream> #include <cstdio>

int main() {

FILE\* file = fopen("non\_existent\_file.txt", "r"); if (file == NULL) {

perror("Error opening file"); return 1;

}

char buffer[100];

if (fread(buffer, 1, 100, file) == 0) { if (ferror(file)) {

perror("Error reading file"); return 1;

}

}

fclose(file); return 0;}



# CHECKING FILE EXISTENCE

#include <iostream> #include <fstream>

int main() {

std::ifstream file("example.txt");

if (!file.is\_open()) {

std::cerr << "Error: File does not exist." << std::endl; return 1;

}

file.close();

return 0;}



# EXCEPTION HANDLING WITH TRY-CATCH

#include <iostream> #include <fstream> int main() {

try {

std::ifstream file("non\_existent\_file.txt"); if (!file.is\_open()) {

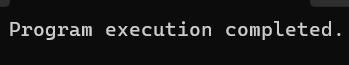
throw std::runtime\_error("Error: File does not exist.");

}

} catch (const std::exception& e) { std::cerr << e.what() << std::endl; return 1;

}

return 0;}



# DETECTING AND HANDLING READ/WRITE ERRORS

#include <iostream> #include <fstream>

int main() {

std::fstream file("example.txt", std::ios::in | std::ios::out | std::ios::trunc);

if (!file.is\_open()) {

std::cerr << "Error: Unable to open file." << std::endl; return 1;

}

// Write to file

file << "Hello, World!" << std::endl; if (file.fail()) {

std::cerr << "Error: Write operation failed." << std::endl; return 1;

}

// Read from file char buffer[100]; file.seekg(0);

file.read(buffer, 100);

if (file.fail()) {

std::cerr << "Error: Read operation failed." << std::endl; return 1;

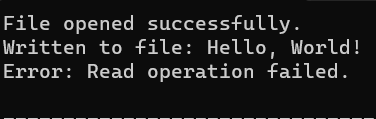
}

file.close();

return 0;

}

***OUTPUT:-***

******