**Problem 1: List Operations**

**Description:**

**Write a program that uses the std::list container to manage a collection of integers. Your program should perform the following operations:**

**Insert elements at the front and back of the list.**

**Remove elements from the front and back of the list.**

**Sort the list in ascending and descending order.**

**Reverse the list.**

**Display the elements of the list.**

#include <iostream>

#include <list>

#include <algorithm>

using namespace std;

int main() {

std::list<int> myList;

// Insert elements at the front and back of the list

myList.push\_front(10);

myList.push\_back(20);

myList.push\_front(5);

myList.push\_back(30);

// Display the elements of the list

cout << "Initial list: ";

for (auto elem : myList) {

cout << elem << " ";

}

cout << endl;

// Remove elements from the front and back of the list

myList.pop\_front();

myList.pop\_back();

// Display the elements of the list

cout << "After removing front and back elements: ";

for (auto elem : myList) {

cout << elem << " ";

}

cout << endl;

// Sort the list in ascending order

myList.sort();

// Display the elements of the list

cout << "Sorted list (ascending): ";

for (auto elem : myList) {

cout << elem << " ";

}

cout <<endl;

// Sort the list in descending order

myList.sort();

myList.reverse();

// Display the elements of the list

cout << "Sorted list (descending): ";

for (auto elem : myList) {

cout << elem << " ";

}

cout << endl;

// Reverse the list

myList.reverse();

// Display the elements of the list

cout << "Reversed list: ";

for (auto elem : myList) {

cout << elem << " ";

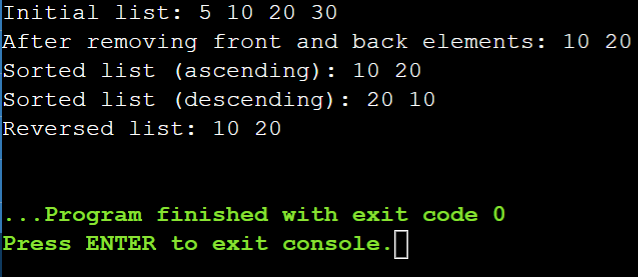
}

cout << endl;

return 0;

}

**Output:**



**Problem 2: Vector Manipulation**

**Description:**

**Create a program that uses the std::vector container to store a collection of floating-point numbers. The program should:**

**Add elements to the vector.**

**Remove elements from a specified position.**

**Find the maximum and minimum elements in the vector.**

**Calculate the average of the elements.**

**Display the elements of the vector.**

#include <iostream>

#include <vector>

#include <algorithm>

#include <numeric>

using namespace std;

int main() {

std::vector<float> vec;

// Add elements to the vector

vec.push\_back(10.5);

vec.push\_back(20.2);

vec.push\_back(5.8);

vec.push\_back(30.1);

vec.push\_back(15.9);

// Display the elements of the vector

cout << "Initial vector: ";

for (std::vector<float>::iterator it = vec.begin(); it != vec.end(); ++it) {

cout << \*it << " ";

}

cout << endl;

// Remove element at position 2

vec.erase(vec.begin() + 2);

// Display the elements of the vector

cout << "After removing element at position 2: ";

for (std::vector<float>::iterator it = vec.begin(); it != vec.end(); ++it) {

cout << \*it << " ";

}

cout << endl;

// Find maximum and minimum elements

float max = \*std::max\_element(vec.begin(), vec.end());

float min = \*std::min\_element(vec.begin(), vec.end());

cout << "Maximum element: " << max << endl;

cout << "Minimum element: " << min << endl;

// Calculate average

float sum = std::accumulate(vec.begin(), vec.end(), 0.0);

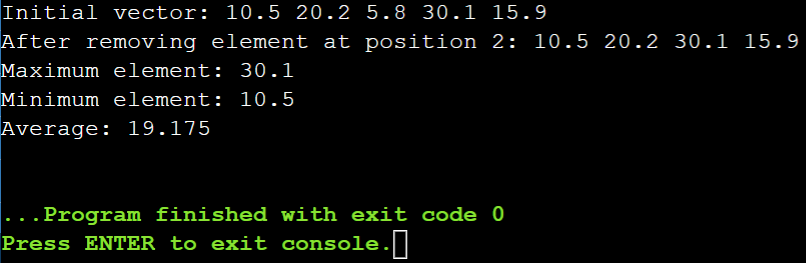
float average = sum / vec.size();

cout << "Average: " << average << endl;

return 0;

}

**Output:**



**Problem 3: Queue Simulation**

**Description:**

**Implement a program using the std::queue container to simulate a ticketing system. The program should:**

**Add customers to the queue.**

**Serve customers (remove from front of the queue).**

**Display the current queue.**

**Display the number of customers served.**

#include <iostream>

#include <queue>

#include <string>

using namespace std;

int main() {

std::queue<std::string> customerQueue;

int customersServed = 0;

// Add customers to the queue

customerQueue.push("Vamsi");

customerQueue.push("Nani");

customerQueue.push("Rakhi");

customerQueue.push("Mani");

customerQueue.push("Sandeep");

// Display the current queue

cout << "Initial Queue: ";

std::queue<std::string> tempQueue = customerQueue;

while (!tempQueue.empty()) {

cout << tempQueue.front() << " ";

tempQueue.pop();

}

cout << endl;

// Serve customers (remove from front of the queue)

while (!customerQueue.empty()) {

cout << "Serving: " << customerQueue.front() << endl;

customerQueue.pop();

customersServed++;

}

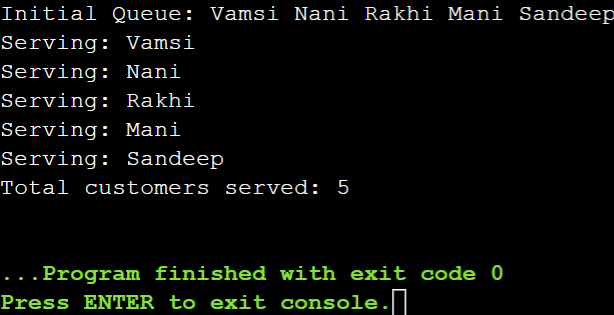
// Display the number of customers served

cout << "Total customers served: " << customersServed << endl;

return 0;

}

**Output:**



**Problem 4: Stack Operations**

**Description:**

**Write a program using the std::stack container to evaluate a postfix expression. The program should:**

**Read a postfix expression.**

**Use a stack to evaluate the expression.**

**Display the result of the evaluation.**

#include <iostream>

#include <stack>

#include <string>

#include <cstdlib>

using namespace std;

int main() {

std::stack<int> evalStack;

std::string postfixExpr;

// Read a postfix expression

cout << "Enter a postfix expression: ";

cin >> postfixExpr;

// Evaluate the expression using a stack

for (char c : postfixExpr) {

if (isdigit(c)) {

evalStack.push(c - '0');

} else {

int operand2 = evalStack.top();

evalStack.pop();

int operand1 = evalStack.top();

evalStack.pop();

if (c == '+') {

evalStack.push(operand1 + operand2);

} else if (c == '-') {

evalStack.push(operand1 - operand2);

} else if (c == '\*') {

evalStack.push(operand1 \* operand2);

} else if (c == '/') {

evalStack.push(operand1 / operand2);

}

}

}

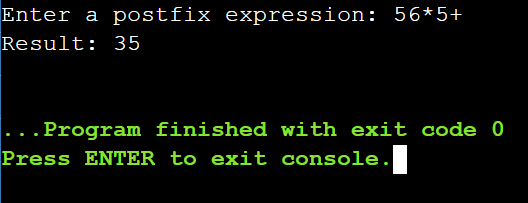
// Display the result of the evaluation

cout << "Result: " << evalStack.top() <<endl;

return 0;

}

**Output:**



#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

string fileName;

char choice;

// Get file name from user

cout << "Enter the file name: ";

cin >> fileName;

// Get user's choice for operation (read or write)

cout << "Enter 'r' to read from the file or 'w' to write to the file: ";

cin >> choice;

if (choice == 'r') {

// Open the file in read mode

ifstream inputFile(fileName);

if (inputFile.is\_open()) {

string line;

// Read data from the file and print line by line

while (getline(inputFile, line)) {

cout << line << endl;

}

inputFile.close();

} else {

cout << "Error opening file for reading." << endl;

}

} else if (choice == 'w') {

// Open the file in write mode (truncates existing content)

ofstream outputFile(fileName);

if (outputFile.is\_open()) {

string content;

// Get content from user to write to the file

cout << "Enter the content to write to the file: ";

getline(cin, content, '\n'); // Include newline character

outputFile << content << endl;

outputFile.close();

cout << "Content written to the file successfully." << endl;

} else {

cout << "Error opening file for writing." << endl;

}

} else {

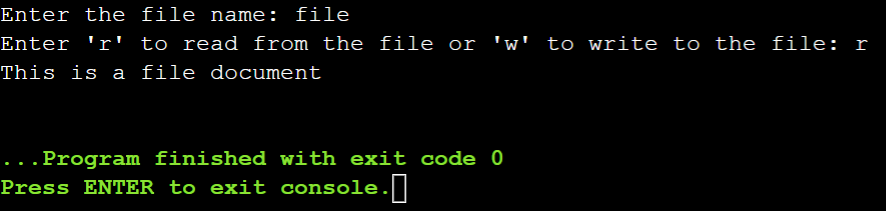
cout << "Invalid choice. Please enter 'r' or 'w'." << endl;

}

return 0;

}

**Output:**



**Problem 1: Read from a File**

**Task:**

**Write a C++ program that reads a text file named input.txt and prints its content to the console.**

**Code:**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

ifstream inputFile("input.txt");

if (inputFile.is\_open()) {

string line;

while (getline(inputFile, line)) {

cout << line << endl;

}

inputFile.close();

} else {

cout << "Error opening file for reading." << endl;

}

return 0;

}

**Questions:**

**How do you open a file for reading in C++?**

Use the ifstream class to create an input file stream and open the file.

**What is the purpose of the ifstream class in C++?**

The ifstream class is used for reading from files. It provides facilities to read data in various formats.

**How can you check if a file was successfully opened?**

Use the is\_open() member function of the ifstream class to check if the file stream has successfully opened the file.

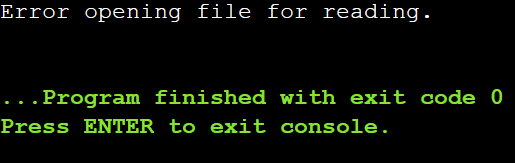
**What function do you use to read a line from a file?**

Use the getline() function to read a line from a file.

**How do you properly close a file after reading?**

Use the close() member function of the ifstream class to close the file.

**Output:**



**Problem 2: Write to a File**

**Task:**

**Write a C++ program that writes the following lines to a file named output.txt:**

**bash**

**Copy code**

**Hello, world!**

**This is a test file.**

**Code:**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

ofstream outputFile("output.txt");

if (outputFile.is\_open()) {

outputFile << "Hello, world!" << endl;

outputFile << "This is a test file." << endl;

outputFile.close();

cout << "Content written to the file successfully." << endl;

} else {

cout << "Error opening file for writing." << endl;

}

return 0;

}

**Questions:**

**How do you open a file for writing in C++?**

Use the ofstream class to create an output file stream and open the file.

**What is the purpose of the ofstream class in C++?**

The ofstream class is used for writing to files. It provides facilities to write data in various formats.

**How can you handle errors if the file fails to open for writing?**

Check if the file stream is successfully opened using the is\_open() member function.

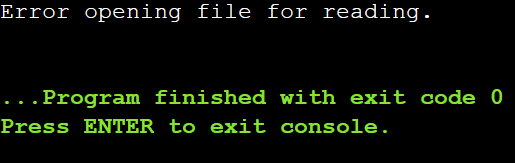
**How do you write a string to a file in C++?**

Use the insertion operator (<<) to write strings to the file.

**What is the importance of closing a file after writing to it?**

Closing a file ensures that all data is properly flushed to the file and resources are released.

**Output:**



**Problem 3: Append to a File**

**Task:**

**Write a C++ program that appends the following line to a file named log.txt:**

**bash**

**Copy code**

**New log entry.**

**Code:**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

ofstream logFile("log.txt", ios::app);

if (logFile.is\_open()) {

logFile << "New log entry." << endl;

logFile.close();

cout << "Content appended to the file successfully." << endl;

} else {

cout << "Error opening file for appending." << endl;

}

return 0;

}

**Questions:**

**How do you open a file for appending in C++?**

Use the ofstream class with the ios::app flag to open the file in append mode.

**What is the difference between opening a file in write mode and append mode?**

Write mode (ios::out) truncates the existing content, while append mode (ios::app) preserves the existing content and adds new content at the end.

**How do you use the ofstream class to append data to a file?**

Open the file using ofstream with the ios::app flag.

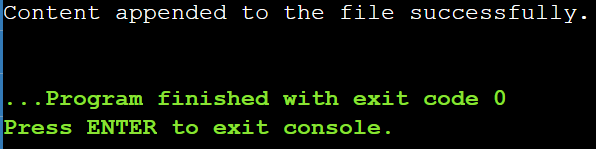
**What happens if the file does not exist when you try to open it in append mode?**

The file will be created if it does not exist.

**How can you ensure data integrity when appending to a file?**

Ensure that the file is successfully opened before appending and properly close the file after the operation.

**Output:**



**Problem 4: Copy a File**

**Task:**

**Write a C++ program that copies the content of a file named source.txt to another file named destination.txt.**

**Code:**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

ifstream sourceFile("source.txt", ios::binary);

ofstream destFile("destination.txt", ios::binary);

if (sourceFile.is\_open() && destFile.is\_open()) {

destFile << sourceFile.rdbuf();

sourceFile.close();

destFile.close();

cout << "File copied successfully." << endl;

} else {

cout << "Error opening source or destination file." << endl;

}

return 0;

}

**Questions:**

**How do you read from one file and write to another file in C++?**

Use ifstream to read from the source file and ofstream to write to the destination file.

**How can you efficiently copy the contents of a file in C++?**

Use the rdbuf() method to copy the file contents in binary mode.

**What are the potential errors you should handle when copying a file?**

Errors in opening the source or destination file and errors during the read/write operations.

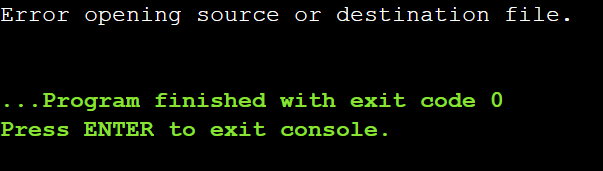
**How do you check the end-of-file (EOF) condition when reading a file?**

The ifstream automatically sets the EOF flag when the end of the file is reached, which can be checked using the eof() member function.

**How do you ensure both files are properly closed after the copy operation?**

Use the close() member function for both ifstream and ofstream objects.

**Output:**



**Problem 5: Count Words in a File**

**Task:**

**Write a C++ program that reads a file named data.txt and counts the number of words in the file.**

**Code:**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

ifstream inputFile("data.txt");

if (inputFile.is\_open()) {

string word;

int wordCount = 0;

while (inputFile >> word) {

wordCount++;

}

inputFile.close();

cout << "Total number of words: " << wordCount << endl;

} else {

cout << "Error opening file for reading." << endl;

}

return 0;

}

**Questions:**

**How do you define a word in the context of reading from a file?**

A word is typically defined as a sequence of characters separated by whitespace (spaces, newlines, etc.).

**What functions can you use to read words from a file in C++?**

Use the extraction operator (>>) to read words from the file.

**How do you handle different word delimiters (spaces, newlines, etc.)?**

The extraction operator (>>) automatically handles spaces, newlines, and other whitespace as delimiters.

**How can you keep track of the word count while reading the file?**

Increment a counter variable each time a word is successfully read.

**How do you handle large files to avoid memory issues while counting words?**

Since only one word is read into memory at a time, large files are handled efficiently without significant memory usage.

**Output:**

