Text Files:

Student Records: Create a program that allows users to enter student information (name, ID, marks) and store them in a text file. The program should allow users to:

Add new student records.

Display all student records from the file.

Search for a specific student by ID and display their details.

Code:

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

const string FILENAME = "student\_records.txt";

void addStudentRecord() {

string name, studentID, marks;

cout << "Enter student's name: ";

cin >> ws;

getline(cin, name);

cout << "Enter student's ID: ";

cin >> studentID;

cout << "Enter student's marks: ";

cin >> marks;

ofstream file(FILENAME, ios::app);

if (file.is\_open()) {

file << name << "," << studentID << "," << marks << endl;

file.close();

cout << "Student record added successfully." << endl;

} else {

cout << "Unable to open file." << endl;

}

}

void displayAllRecords() {

ifstream file(FILENAME);

string line;

if (file.is\_open()) {

cout << "Student Records:" << endl;

while (getline(file, line)) {

cout << line << endl;

}

file.close();

} else {

cout << "No student records found." << endl;

}

}

void searchStudentByID() {

string studentID, line;

bool found = false;

cout << "Enter student's ID to search: ";

cin >> studentID;

ifstream file(FILENAME);

if (file.is\_open()) {

while (getline(file, line)) {

size\_t pos = line.find(",");

string id = line.substr(pos + 1, line.find(",", pos + 1) - pos - 1);

if (id == studentID) {

cout << "Student found: " << line << endl;

found = true;

break;

}

}

file.close();

if (!found) {

cout << "Student with ID " << studentID << " not found." << endl;

}

} else {

cout << "No student records found." << endl;

}

}

int main() {

int choice;

while (true) {

cout << "1. Add Student Record" << endl;

cout << "2. Display All Records" << endl;

cout << "3. Search Student by ID" << endl;

cout << "4. Exit" << endl;

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

addStudentRecord();

break;

case 2:

displayAllRecords();

break;

case 3:

searchStudentByID();

break;

case 4:

cout << "Exiting..." << endl;

return 0;

default:

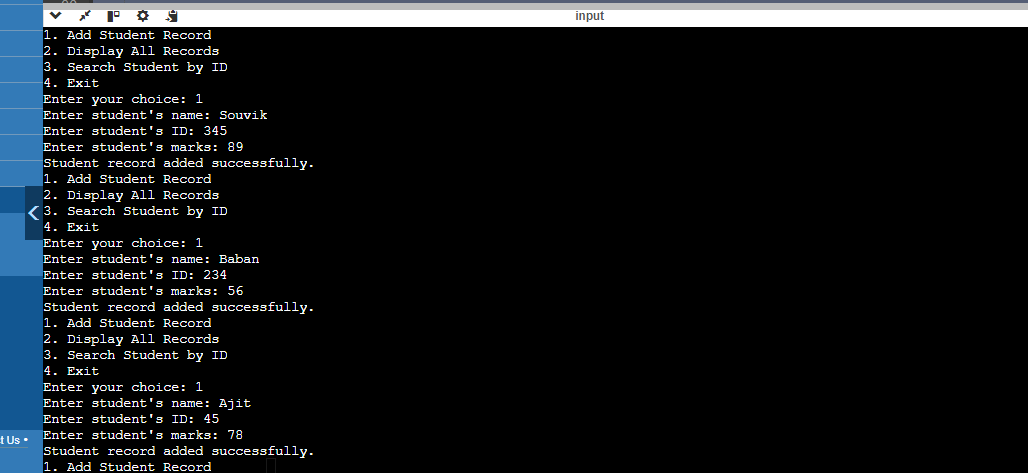
cout << "Invalid choice. Please try again." << endl;

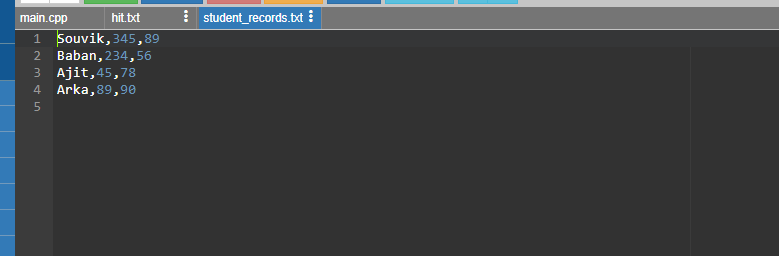
}

}

}

Output:





Phonebook: Develop a program that functions as a simple phonebook. Users can:

Add new contacts (name, phone number) to the file.

Search for a contact by name and display their phone number.

File Encryption/Decryption (Optional): Implement a program that encrypts/decrypts a text file using a simple Caesar cipher or another basic encryption method.

Code:

#include <iostream>

#include <fstream>

#include <string>

#include <map>

using namespace std;

string encrypt(const string& str, int shift) {

string encrypted;

for (char c : str) {

if (c >= 'a' && c <= 'z') {

encrypted += 'a' + (c - 'a' + shift) % 26;

} else if (c >= 'A' && c <= 'Z') {

encrypted += 'A' + (c - 'A' + shift) % 26;

} else {

encrypted += c;

}

}

return encrypted;

}

string decrypt(const string& str, int shift) {

return encrypt(str, 26 - shift);

}

void addContact(map<string, string>& phonebook, const string& name, const string& number) {

phonebook[name] = number;

}

void searchContact(const map<string, string>& phonebook, const string& name) {

auto it = phonebook.find(name);

if (it != phonebook.end()) {

cout << "Phone number: " << it->second << endl;

} else {

cout << "Contact not found." << endl;

}

}

void savePhonebook(const map<string, string>& phonebook, const string& filename, int shift) {

ofstream file(filename);

if (file.is\_open()) {

for (const auto& entry : phonebook) {

string encryptedName = encrypt(entry.first, shift);

string encryptedNumber = encrypt(entry.second, shift);

file << encryptedName << ":" << encryptedNumber << endl;

}

file.close();

} else {

cout << "Unable to open file." << endl;

}

}

void loadPhonebook(map<string, string>& phonebook, const string& filename, int shift) {

ifstream file(filename);

if (file.is\_open()) {

string line;

while (getline(file, line)) {

size\_t colonPos = line.find(':');

if (colonPos != string::npos) {

string encryptedName = line.substr(0, colonPos);

string encryptedNumber = line.substr(colonPos + 1);

string name = decrypt(encryptedName, shift);

string number = decrypt(encryptedNumber, shift);

phonebook[name] = number;

}

}

file.close();

} else {

cout << "Unable to open file." << endl;

}

}

int main() {

const string filename = "phonebook.txt";

const int shift = 3;

map<string, string> phonebook;

loadPhonebook(phonebook, filename, shift);

while (true) {

cout << "Phonebook Menu:" << endl;

cout << "1. Add contact" << endl;

cout << "2. Search contact" << endl;

cout << "3. Save and exit" << endl;

int choice;

cin >> choice;

if (choice == 1) {

string name, number;

cout << "Enter name: ";

cin >> name;

cout << "Enter phone number: ";

cin >> number;

addContact(phonebook, name, number);

} else if (choice == 2) {

string name;

cout << "Enter name: ";

cin >> name;

searchContact(phonebook, name);

} else if (choice == 3) {

savePhonebook(phonebook, filename, shift);

return 0;

} else {

cout << "Invalid choice." << endl;

}

}

return 0;

}

Output:

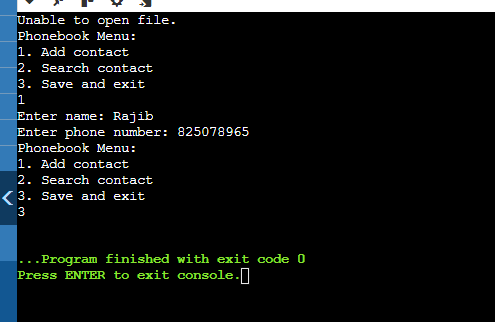


Image Copy: Write a program that copies the contents of an image file (e.g., JPG, PNG) to a new file. Ensure you handle binary data correctly.

Code:

#include <iostream>

#include <fstream>

using namespace std;

void copyImage(const std::string& inputFilePath, const std::string& outputFilePath) {

ifstream inputFile(inputFilePath, ios::binary);

if (!inputFile) {

cerr << "File not found: " << inputFilePath << endl;

return;

}

ofstream outputFile(outputFilePath, ios::binary);

if (!outputFile) {

cerr << "Could not open output file: " << outputFilePath << endl;

return;

}

outputFile << inputFile.rdbuf();

cout << "Image copied successfully from " << inputFilePath << " to " << outputFilePath << endl;

}

int main() {

string inputFilePath = "path/to/input/image.jpg";

string outputFilePath = "path/to/output/image\_copy.jpg";

copyImage(inputFilePath, outputFilePath);

return 0;

}

Output:



Inventory Management: Develop a program that manages a store inventory. Users can:

Add new items (name, price, quantity) to a binary file.

Display all items from the inventory.

Update the quantity of an existing item.

High Score Tracking (Optional): Create a program that keeps track of high scores for a game. Users can:

Save a new high score to a binary file.

Display the current high score.

Code:

#include <iostream>

#include <fstream>

#include <vector>

#include <string>

using namespace std;

struct Item {

string name;

float price;

int quantity;

void save(std::ofstream& out) const {

size\_t nameLength = name.size();

out.write(reinterpret\_cast<const char\*>(&nameLength), sizeof(nameLength));

out.write(name.c\_str(), nameLength);

out.write(reinterpret\_cast<const char\*>(&price), sizeof(price));

out.write(reinterpret\_cast<const char\*>(&quantity), sizeof(quantity));

}

void load(ifstream& in) {

size\_t nameLength;

in.read(reinterpret\_cast<char\*>(&nameLength), sizeof(nameLength));

name.resize(nameLength);

in.read(&name[0], nameLength);

in.read(reinterpret\_cast<char\*>(&price), sizeof(price));

in.read(reinterpret\_cast<char\*>(&quantity), sizeof(quantity));

}

};

void addItem(const string& filename) {

Item item;

cout << "Enter item name: ";

cin >> item.name;

cout << "Enter item price: ";

cin >> item.price;

cout << "Enter item quantity: ";

cin >> item.quantity;

ofstream outFile(filename, ios::binary | ios::app);

if (outFile.is\_open()) {

item.save(outFile);

outFile.close();

cout << "Item added successfully.\n";

} else {

cerr << "Error opening file for writing.\n";

}

}

void displayItems(const std::string& filename) {

ifstream inFile(filename, ios::binary);

if (inFile.is\_open()) {

Item item;

while (inFile.peek() != EOF) {

item.load(inFile);

cout << "Name: " << item.name << ", Price: " << item.price << ", Quantity: " << item.quantity << "\n";

}

inFile.close();

} else {

cerr << "Error opening file for reading.\n";

}

}

void updateQuantity(const string& filename, const string& itemName, int newQuantity) {

ifstream inFile(filename, ios::binary);

vector<Item> items;

if (inFile.is\_open()) {

Item item;

while (inFile.peek() != EOF) {

item.load(inFile);

items.push\_back(item);

}

inFile.close();

bool itemFound = false;

for (auto& i : items) {

if (i.name == itemName) {

i.quantity = newQuantity;

itemFound = true;

break;

}

}

if (itemFound) {

ofstream outFile(filename, ios::binary | ios::trunc);

if (outFile.is\_open()) {

for (const auto& i : items) {

i.save(outFile);

}

outFile.close();

cout << "Quantity updated successfully.\n";

} else {

cerr << "Error opening file for writing.\n";

}

} else {

cout << "Item not found.\n";

}

} else {

cerr << "Error opening file for reading.\n";

}

}

int main() {

string filename = "inventory.dat";

int choice;

string itemName;

int newQuantity;

while (true) {

cout << "1. Add Item\n2. Display Items\n3. Update Quantity\n4. Exit\nEnter your choice: ";

cin >> choice;

switch (choice) {

case 1:

addItem(filename);

break;

case 2:

displayItems(filename);

break;

case 3:

cout << "Enter item name to update: ";

cin >> itemName;

cout << "Enter new quantity: ";

cin >> newQuantity;

updateQuantity(filename, itemName, newQuantity);

break;

case 4:

return 0;

default:

cout << "Invalid choice. Try again.\n";

}

}

}

Output:

