One-Week-C-Primer (Day 7)

Day 7: Mini Project

Note : Refer to the Attached Document along with this for the codes

Advanced File Encryption and Decryption Program

In this advanced program, we will create a file encryption and decryption utility using symmetric key encryption (AES) from the OpenSSL library. The user can encrypt a file with a specified password and later decrypt it back to its original form.

Theory:

We will use the Advanced Encryption Standard (AES) algorithm, a widely used symmetric key encryption algorithm, to perform the encryption and decryption. The program will utilize the OpenSSL library to access AES functions for encryption and decryption. The user will provide the input file, password, and the output file name for encryption and decryption operations.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <openssl/aes.h>

#define AES_KEY_SIZE 128 // AES key size in bits (128, 192, or 256)

void encryptFile(const char *inputFileName, const char *outputFileName, const char *passwo
rd) {
    unsigned char iv[AES_BLOCK_SIZE]; // Initialization Vector (IV)
    AES_KEY aesKey;

    FILE *inputFile = fopen(inputFileName, "rb");
    FILE *outputFile = fopen(outputFileName, "wb");

    if (!inputFile || !outputFile) {
        printf("Error opening files.\\n");
        return;
    }

    // Generate random IV
    if (RAND_bytes(iv, AES_BLOCK_SIZE) != 1) {
        printf("Error generating IV.\\n");
```

```
        fclose(inputFile);
        fclose(outputFile);
        return;
    }

    // Set up AES encryption key
    if (AES_set_encrypt_key((const unsigned char *)password, AES_KEY_SIZE, &aesKey) != 0)
 {
        printf("Error setting up AES encryption key.\\n");
        fclose(inputFile);
        fclose(outputFile);
        return;
    }

    // Write the IV to the output file
    fwrite(iv, 1, AES_BLOCK_SIZE, outputFile);

    int bytesRead, bytesEncrypted;
    unsigned char inputBuffer[AES_BLOCK_SIZE];
    unsigned char outputBuffer[AES_BLOCK_SIZE];

    // Perform encryption on each block of data
    while ((bytesRead = fread(inputBuffer, 1, AES_BLOCK_SIZE, inputFile)) > 0) {
        AES_cfb128_encrypt(inputBuffer, outputBuffer, bytesRead, &aesKey, iv, &bytesEncryp
ted, AES_ENCRYPT);
        fwrite(outputBuffer, 1, bytesEncrypted, outputFile);
    }

    fclose(inputFile);
    fclose(outputFile);

    printf("File encrypted successfully.\\n");
}

void decryptFile(const char *inputFileName, const char *outputFileName, const char *passwo
rd) {
    unsigned char iv[AES_BLOCK_SIZE]; // Initialization Vector (IV)
    AES_KEY aesKey;

    FILE *inputFile = fopen(inputFileName, "rb");
    FILE *outputFile = fopen(outputFileName, "wb");

    if (!inputFile || !outputFile) {
        printf("Error opening files.\\n");
        return;
    }

    // Read IV from the input file
    if (fread(iv, 1, AES_BLOCK_SIZE, inputFile) != AES_BLOCK_SIZE) {
        printf("Error reading IV from the input file.\\n");
        fclose(inputFile);
        fclose(outputFile);
        return;
    }
```

```
    // Set up AES decryption key
    if (AES_set_decrypt_key((const unsigned char *)password, AES_KEY_SIZE, &aesKey) != 0)
 {
        printf("Error setting up AES decryption key.\\n");
        fclose(inputFile);
        fclose(outputFile);
        return;
    }

    int bytesRead, bytesDecrypted;
    unsigned char inputBuffer[AES_BLOCK_SIZE];
    unsigned char outputBuffer[AES_BLOCK_SIZE];

    // Perform decryption on each block of data
    while ((bytesRead = fread(inputBuffer, 1, AES_BLOCK_SIZE, inputFile)) > 0) {
        AES_cfb128_encrypt(inputBuffer, outputBuffer, bytesRead, &aesKey, iv, &bytesDecryp
ted, AES_DECRYPT);
        fwrite(outputBuffer, 1, bytesDecrypted, outputFile);
    }

    fclose(inputFile);
    fclose(outputFile);

    printf("File decrypted successfully.\\n");
}

int main() {
    char inputFileName[256];
    char outputFileName[256];
    char password[256];
    int choice;

    printf("File Encryption and Decryption Program\\n");
    printf("1. Encrypt File\\n");
    printf("2. Decrypt File\\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    printf("Enter input file name: ");
    scanf("%s", inputFileName);

    printf("Enter output file name: ");
    scanf("%s", outputFileName);

    printf("Enter password: ");
    scanf("%s", password);

    if (choice == 1) {
        encryptFile(inputFileName, outputFileName, password);
    } else if (choice == 2) {
        decryptFile(inputFileName, outputFileName, password);
    } else {
        printf("Invalid choice.\\n");
```

```
    }

    return 0;
}
```

Instructions:

1. Compile the code with OpenSSL library.

   ```
   gcc -o file_encrypt_decrypt file_encrypt_decrypt.c -lcrypto
   ```

2. Run the compiled program.
3. Choose option 1 to encrypt a file or option 2 to decrypt a file.
4. Enter the input file name, output file name, and password.
5. The program will perform encryption or decryption based on the chosen option and display a success message.

Note:

1. Make sure you have OpenSSL installed on your system before compiling the code with the `lcrypto` flag.
2. This program uses AES encryption for demonstration purposes. In real-world applications, it is essential to use proper key management and security practices. Additionally, consider using more advanced encryption libraries like OpenSSL or libsodium for robust encryption and security.

Simple Address Book:

In this program, we will create a simple text-based address book that allows users to store and manage contact information. Users can add new contacts, view existing contacts, search for contacts, and delete contacts.

Theory:

For this address book, we will use a structure to represent each contact, and an array of structures to store multiple contacts. The user will interact with the program through a menu-driven

interface. The program will use file handling to store the contacts in a text file, enabling data persistence between program runs.

Code:

```
#include <stdio.h>
#include <string.h>

#define MAX_CONTACTS 100

// Structure to represent a contact
struct Contact {
    char name[50];
    char phone[15];
    char email[50];
};

// Function to add a new contact
void addContact(struct Contact contacts[], int *numContacts) {
    if (*numContacts >= MAX_CONTACTS) {
        printf("Address book is full. Cannot add more contacts.\\n");
        return;
    }

    struct Contact newContact;

    printf("Enter name: ");
    scanf("%s", newContact.name);
    printf("Enter phone: ");
    scanf("%s", newContact.phone);
    printf("Enter email: ");
    scanf("%s", newContact.email);

    contacts[*numContacts] = newContact;
    (*numContacts)++;

    printf("Contact added successfully.\\n");
}

// Function to view all contacts
void viewContacts(struct Contact contacts[], int numContacts) {
    if (numContacts == 0) {
        printf("Address book is empty.\\n");
        return;
    }

    printf("Contact List:\\n");
    for (int i = 0; i < numContacts; i++) {
        printf("Name: %s\\n", contacts[i].name);
        printf("Phone: %s\\n", contacts[i].phone);
        printf("Email: %s\\n\\n", contacts[i].email);
```

```
    }
}

// Function to search for a contact by name
void searchContact(struct Contact contacts[], int numContacts) {
    char searchName[50];
    int found = 0;

    printf("Enter name to search: ");
    scanf("%s", searchName);

    for (int i = 0; i < numContacts; i++) {
        if (strcmp(contacts[i].name, searchName) == 0) {
            printf("Contact found:\\n");
            printf("Name: %s\\n", contacts[i].name);
            printf("Phone: %s\\n", contacts[i].phone);
            printf("Email: %s\\n\\n", contacts[i].email);
            found = 1;
            break;
        }
    }

    if (!found) {
        printf("Contact not found.\\n");
    }
}

// Function to delete a contact by name
void deleteContact(struct Contact contacts[], int *numContacts) {
    char deleteName[50];
    int found = 0;

    printf("Enter name to delete: ");
    scanf("%s", deleteName);

    for (int i = 0; i < *numContacts; i++) {
        if (strcmp(contacts[i].name, deleteName) == 0) {
            for (int j = i; j < *numContacts - 1; j++) {
                contacts[j] = contacts[j + 1];
            }
            (*numContacts)--;
            printf("Contact deleted successfully.\\n");
            found = 1;
            break;
        }
    }

    if (!found) {
        printf("Contact not found.\\n");
    }
}

int main() {
    struct Contact contacts[MAX_CONTACTS];
```

```
    int numContacts = 0;
    int choice;

    while (1) {
        printf("\\nSimple Address Book\\n");
        printf("1. Add Contact\\n");
        printf("2. View Contacts\\n");
        printf("3. Search Contact\\n");
        printf("4. Delete Contact\\n");
        printf("5. Exit\\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                addContact(contacts, &numContacts);
                break;
            case 2:
                viewContacts(contacts, numContacts);
                break;
            case 3:
                searchContact(contacts, numContacts);
                break;
            case 4:
                deleteContact(contacts, &numContacts);
                break;
            case 5:
                printf("Exiting the program.\\n");
                return 0;
            default:
                printf("Invalid choice. Please try again.\\n");
        }
    }

    return 0;
}
```

Text-Based Game: The Dungeon Adventure

In this advanced text-based game, we will create a "Dungeon Adventure" where the player explores a mysterious dungeon, encounters monsters, finds treasures, and makes choices to progress through the game. The goal is to reach the end of the dungeon and escape safely.

Theory:

The game will use random number generation to determine encounters with monsters and treasure finds. The player will have attributes like health, attack power, and defense, which will

change based on their decisions and outcomes in the game. The player will be able to navigate through different rooms in the dungeon and make choices during encounters.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <stdbool.h>

// Function to generate a random number between min and max (inclusive)
int getRandomNumber(int min, int max) {
    return rand() % (max - min + 1) + min;
}

// Function to simulate a battle with a monster
bool battle(int *playerHealth, int *playerAttack, int *playerDefense) {
    int monsterHealth = getRandomNumber(50, 100);
    int monsterAttack = getRandomNumber(10, 20);
    int monsterDefense = getRandomNumber(5, 10);

    printf("You encountered a fearsome monster!\n");

    while (*playerHealth > 0 && monsterHealth > 0) {
        int playerDamage = *playerAttack - monsterDefense;
        int monsterDamage = monsterAttack - *playerDefense;

        if (playerDamage > 0) {
            monsterHealth -= playerDamage;
        } else {
            playerDamage = 0;
        }

        if (monsterDamage > 0) {
            *playerHealth -= monsterDamage;
        } else {
            monsterDamage = 0;
        }

        printf("You dealt %d damage to the monster. Monster's health: %d\n", playerDamage,
monsterHealth);
        printf("Monster dealt %d damage to you. Your health: %d\n", monsterDamage, *player
Health);
    }

    if (*playerHealth <= 0) {
        printf("You were defeated by the monster. Game Over!\n");
        return false;
    } else {
        printf("Congratulations! You defeated the monster!\n");
        return true;
```

```
    }
}

int main() {
    srand(time(NULL));
    int playerHealth = 100;
    int playerAttack = 20;
    int playerDefense = 10;

    printf("Welcome to the Dungeon Adventure!\n");
    printf("You are in a dark and mysterious dungeon. Your goal is to reach the exit safel
y.\n");

    while (playerHealth > 0) {
        printf("\nYou are in a new room. What do you want to do?\n");
        printf("1. Explore the room\n");
        printf("2. Rest and recover health\n");
        printf("3. Try to find an exit\n");
        printf("Enter your choice: ");

        int choice;
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                if (getRandomNumber(1, 2) == 1) {
                    if (battle(&playerHealth, &playerAttack, &playerDefense) == false) {
                        return 0;
                    }
                } else {
                    int treasure = getRandomNumber(20, 50);
                    printf("You found a treasure chest containing %d gold coins!\n", treas
ure);
                }
                break;
            case 2:
                playerHealth += getRandomNumber(10, 20);
                printf("You rested and recovered some health. Your health: %d\n", playerHe
alth);
                break;
            case 3:
                if (getRandomNumber(1, 2) == 1) {
                    printf("Congratulations! You found the exit and escaped the dungeon sa
fely!\n");
                    return 0;
                } else {
                    printf("You searched but couldn't find the exit in this room.\n");
                }
                break;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }
```

```
    printf("Your health has reached zero. Game Over!\n");

    return 0;
}
```

Instructions:

1. Compile and run the code.
2. You are in a mysterious dungeon. Your goal is to reach the exit safely.
3. During your adventure, you will encounter monsters and treasure chests.
4. You have three choices in each room: explore the room, rest and recover health, or try to find the exit.
5. Exploring the room may lead to a battle with a monster or finding a treasure chest.
6. Resting will recover some of your health.
7. Trying to find the exit may or may not lead to success.
8. The game ends if your health reaches zero or if you find the exit safely.
