C++프로그래밍및실습

암호화 프로그램

진척 보고서 #1

제출일자: 2024/11/17
제출자명: 정철원
제출자학번: 214771

1. 프로젝트 목표

1) 배경 및 필요성

데이터 보안과 개인 정보 보호가 중요한 이슈가 되면서, 정보를 안전하게 전송하고 보관하기 위한 암호화의 필요성이 높아지고 있습니다. 개인이나 기업이 데이터를 암호화하지 않고 전송하거나 저장할 경우, 해킹이나 데이터 유출로 인한 피해가 발생할 수 있습니다. 특히, 데이터를 파일 형태로 저장할 때 이를 암호화하여 보호하는 방법은 필수적입니다. 이 문제를 해결하기 위해 사용자들이 손쉽게 사용할 수 있는 간단한 암호화 프로그램이 필요합니다.

2) 프로젝트 목표

사용자가 입력한 텍스트를 암호화하여 파일에 저장하고, 이 파일에 저장된 데이터를 복호화하여 원본 데이터로 복원하는 프로그램을 구현하는 것을 목표로 합니다. 간단한 암호화 및 복호화 기능을 제공하여, 데이터 보안을 강화하고 사용자가 정보를 안전하게 전달할 수 있도록 돕습니다.

3) 차별점

일반적으로 복잡한 암호화 소프트웨어는 사용하기 어렵고 설정이 복잡한 경우가 많습니다. 본 프로그램은 누구나 쉽게 사용할 수 있도록 단순한 사용 절차를 제공하며, 입문자도 쉽게 암호화/복호화 기능을 이용할 수 있는 것이 특징입니다.

2. 기능 계획

1) 기능 1 : 기본 암호화 및 복호화 구현

- 설명 : 사용자가 입력한 텍스트를 일정한 규칙(예: 시저 암호)을 이용해 암호화하고 복호화하는 기능입니다. 특정 시프트 값을 입력하면, 각 문자가 해당 시프트 값만큼 이동하여 암호화되고, 암호화에 사용된 시프트 값을 입력하면, 암호화된 텍스트를 다시 원본으로 복호화할 수 있습니다.

(1) 세부 기능 1 : encryptCaesar 함수

- 설명 : 입력된 텍스트를 암호화합니다.

(2) 세부 기능 2 : decryptCaesar 함수

- 설명 : 암호화된 텍스트를 복호화합니다.

2) 기능 2 : 암호화된 파일의 다중 텍스트 저장

- 설명 : 파일에 여러 줄의 암호화된 텍스트를 저장할 수 있도록 기능을 확장합니다.

(1) 세부 기능 1 : 암호화된 데이터를 파일에 추가로 저장

- 설명 : 파일에 암호화된 데이터를 추가로 저장 가능합니다.

(2) 세부 기능 2 : 여러 줄의 데이터를 읽고 복호화하는 기능

- 설명 : 복호화 시 모든 데이터를 처리할 수 있습니다.

3) 기능 3 : 암호화 강도 조절 (랜덤 시프트 값)

- 설명 : 암호화 강도를 높이기 위해 랜덤한 시프트 값을 생성하여 데이터를 암호화합니다.

(1) 세부 기능 1 : 무작위 시프트 값을 생성하여 암호화하고 복호화에 사용할 수 있도록 함

- 설명 : 랜덤 시프트 값으로 강력한 암호화가 가능합니다.

4) 기능 4 : 암호화 파일 보호 (비밀번호 설정 및 확인)

- 설명 : 파일 접근 전 비밀번호를 설정하고 확인하여 보안을 강화합니다.

(1) 세부 기능 1 : 사용자가 비밀번호를 설정

- 설명 : 비밀번호 보호 기능을 통해 파일 접근 보안을 강화한다.

5) 기능 5 : 암호화된 데이터 삭제 기능

- 설명 : 파일에 저장된 암호화 데이터를 삭제하여 민감 정보를 안전하게 처리합니다.

(1) 세부 기능 1 : 암호화된 데이터를 저장한 파일 삭제

- 설명 : 암호화 데이터를 안전하게 삭제하여 민감 정보를 보호 가능합니다.

3. 진척사항

1) 기능 구현

(1) encryptCaesar 함수

```
// 시저 암호로 텍스트를 암호화하는 함수
string encryptCaesar(const string &text, int shift) {
    string encryptedText = text;
    for (char &c : encryptedText) {
        if (isalpha(c)) {  // 알파벳만 암호화
            char base = isupper(c) ? 'A' : 'a';
            c = (c - base + shift) % 26 + base;
        }
    }
    return encryptedText;
}
```

입출력

- const string &text = 암호화할 문자열, 띄어쓰기와 숫자는 암호화되지 않으며, 영어 알파벳만 암호화됨
- int shift = 알파벳을 시피트할 값, shift가 3일 경우, 'A'는 'D로 'Z'는 'C'로 암호화됨
- encryptedText = 암호화된 결과를 저장할 변수
- base = 기준이 되는 위치를 저장 (A 또는 a를 저장)

반환값

- 암호화된 문자열을 반환

결과

- 사용자가 입력한 문자열에서 영어 알파벳 문자는 입력한 시프트 값만큼 이동한 결과를 반환

설명

- for루프를 통해 문자열의 각 문자를 순회하고 isalpha 함수를 통해 문자가 알파벳인지 확인(알파벳만 암호화 대상임)

- isupper 함수를 통해 대소문자를 구분하고 삼항 연산자로 기준 문자를 저장함

- (c- base + shift) % 26 + base에서 c에서 base를 빼 문자의 상대적 위치를 구한 뒤 shift를 더해 26으로 나누어 알파벳의 범위를 유지함, 다시 base를 더해 시프트된 문자 값을 얻음

적용된 배운 내용

- 4주차(조건문 반복문), 6주차(함수), 7주차(문자열)

(2) decryptCaesar 함수

```
// 시저 암호로 텍스트를 복호화하는 함수
string decryptCaesar(const string &text, int shift) {
    return encryptCaesar(text, 26 - shift);  // 복호화는 암호화의 반대
}
```

입출력

- const string &text : 복호화할 암호화된 문자열

- int shift : 암호화 시 사용된 시프트 값

반환값

- 복호화된 문자열을 반환

결과

- 시저 암호로 암호화된 문자열에서 시프트 값을 반대로 적용하여 원래 텍스트를 복구함

설명

- 암호화 시 사용한 시프트 값 shift를 26에서 뺀 값을 사용해 암호화와 동일한 방식으로 처리하면 복호화가 됨. 내부적으로 encryptCaesar 함수를 재사용하여, 복호화와 암호화를 동일한 방식으로 처리

적용된 배운내용

- 6주차(함수), 7주차(문자열)

(3) main 함수

1. 암호화 모드

```
int main() {
    int mode;
    cout << "모드를 선택하세요 (1: 암호화, 2: 복호화): ";
    cin >> mode;
    cin.ignore();  // 입력 버퍼 해우기

    if (mode == 1) {
        // 암호화 모드
        string text;
        int shift;
        cout << "암호화할 텍스트를 입력하세요: ";
        getline(cin, text);  // 띄어쓰기를 포함한 암호화할 텍스트 입력하기
        cout << "시프트 값을 입력하세요 (1~25): "; // 알파벳은 총 26자리가 있기때문에 1~25숫자만 가능 26 입력 시 동일한 텍스트가 나옴
        cin >> shift;  // 시프트 값 입력

        // 암호화 및 출력
        string encryptedText = encryptCaesar(text, shift);
        cout << "암호화된 텍스트: " << encryptedText << endl;
```

입출력

- mode : 암호화 복호화 모드를 선택하는 정수
- text : 암호화 또는 복호화 대상 텍스트
- shift : 알파벳 이동 시 사용할 시프트 값
- encryptedText = 암호화된 결과를 저장할 변수

결과

- 사용자가 입력한 텍스트를 시저 암호 방식으로 암호화하고 결과를 출력

설명

- 사용자가 1(암호화) 또는 2(복호화)를 입력함
- cin.ignore()를 호출하여, 이전에 입력받은 mode의 버퍼를 비움. 그렇지 않으면 다음 입력(getline)이 제대로 작동하지 않음
- 암호화할 텍스트와 시프트 값을 입력받음
- encryptCaesar 함수를 호출하여 암호화 결과를 반환받아 출력

적용된 배운 내용

- 조건문(4주차), 함수(6주차)

2. 복호화 모드

```
    } else if (mode == 2) {
        // 복호화 모드
        string encryptedText;
        int shift;
        cout << "복호화할 텍스트를 입력하세요: ";
        getline(cin, encryptedText);  // 암호화된 텍스트 입력
        cout << "시프트 값을 입력하세요 (1~25): ";
        cin >> shift;  // 시프트 값 입력

        // 복호화 및 출력
        string decryptedText = decryptCaesar(encryptedText, shift);
        cout << "복호화된 텍스트: " << decryptedText << endl;
    } else {
        cout << "잘못된 선택입니다." << endl;
    }

    return 0;
}
```

입출력

- mode : 암호화 복호화 모드를 선택하는 정수
- shift : 알파벳 이동 시 사용할 시프트 값
- encryptedText = 암호화된 텍스트를 입력하는 변수
- decryptedText = 복호화된 결과를 저장할 변수

결과

- 암호화된 텍스트를 복호화하여 원래 텍스트를 출력

설명

- 복호화할 텍스트와 시프트 값을 입력받음
- decryptCaesar 함수를 호출하여 복호화 결과를 반환받아 출력
- mode가 1, 2 이외의 값일 경우, "잘못된 선택입니다."를 출력하고 프로그램을 종료

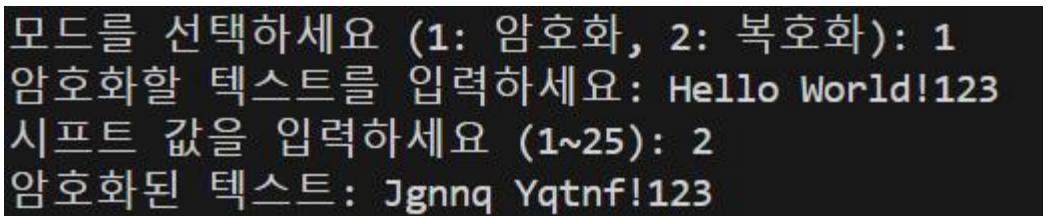
적용된 배운 내용

- 조건문(4주차), 함수(6주차)

2) 테스트 결과

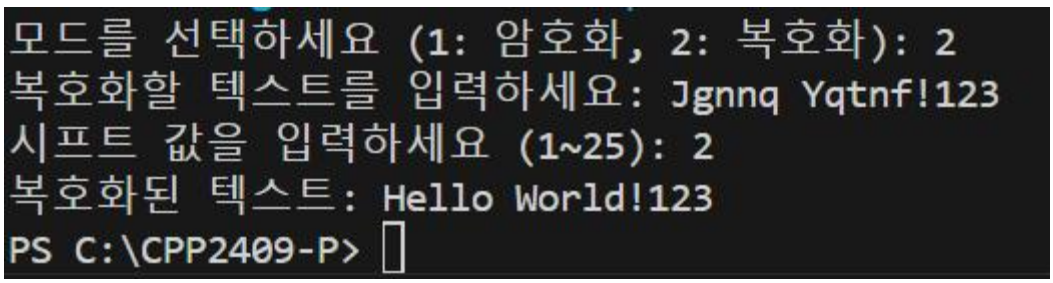
(1) 암호화 기능

대소문자를 구문하고 공백, 특수문자, 숫자는 암호화 하지 않음

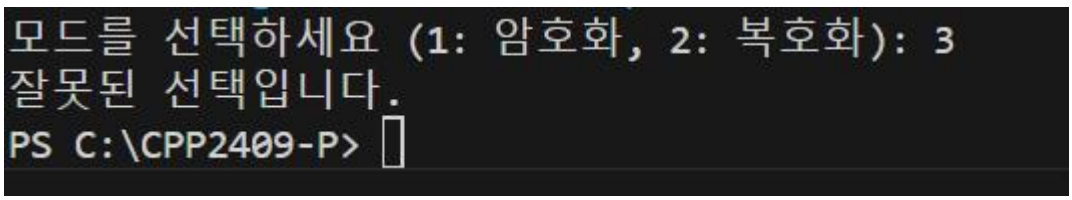


(2) 복호화 기능

대소문자를 구문하고 공백, 특수문자, 숫자는 복호화 하지 않음



(3) mode 1,2를 제외한 다른 값이 들어왔을때 오류 표시 기능



4. 계획 대비 변경 사항

1) 전체적인 기능 여럿 추가

- 이전 : 기존에는 암호화, 복호화, 파일 저장 기능 밖에 없었다

- 이후 : 암호화 강도 조절, 암호화 파일 비밀번호 설정, 암호화 파일 삭제 기능을 추가하였다.

- 사유 : 암호화, 복호화 기능들로만은 보안 프로그램의 세밀성을 전부 표현할 순 없을거 같고, 프로젝트를 진행하다보니 다양한 기능을 구현해보고 싶은 욕심이 생겼기 때문이다.

5. 프로젝트 일정

업무		11/3	11/17	11/24	12/1
제안서 작성		완료			
기능1	세부기능1,2	완료			
기능2	세부기능1		진행중		
	세부기능2			---------->	

업무		12/1	12/8	12/15	12/22
기능3	세부기능1	---------->			
기능4	세부기능1		---------->		
기능5	세부기능1			---------->	