Classic Cipher 실습 보고서

정보컴퓨터공학부 201924437 김윤하

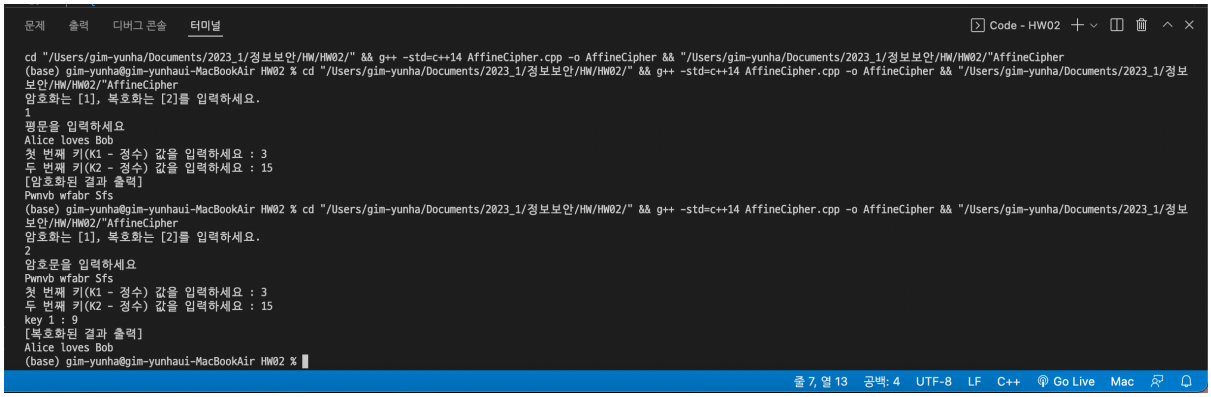
[문제 1] 강의 자료 8p의 Affine cipther 소스 코드를 작성하고 동작 과정을 기술하시오.

[문제 1 소스 코드]

```
#include <iostream>
#include <cstring>
using namespace std;

char *AffineCeasarCipher(int, char *, int, int, int);

int main() {
    int key1, key2, str_size=0;
    char str[50] = {0, };

    int mode;
    cout << "암호화는 [1], 복호화는 [2]를 입력하세요." << endl;
    cin >> mode;
    cin.ignore();
    switch (mode)
    {
    case 1:
    {
        cout << "평문을 입력하세요" << endl;
        cin.getline(str, sizeof(str));
        break;
    }
    case 2:
    {
        cout << "암호문을 입력하세요" << endl;
        cin.getline(str, sizeof(str));
        break;
    }
    default:
    {
        cout << "[INPUT ERROR] 제대로 입력하지 않았습니다." << endl;
        return 0;
    }
    }

    cout << "첫 번째 키(K1 - 정수) 값을 입력하세요 : ";
    cin >> key1;
    cout << "두 번째 키(K2 - 정수) 값을 입력하세요 : ";
    cin >> key2;
```

```
    // 복호화를 위해 key1의 inverse 값 구하기
    if (mode == 2) {
        int flag = 0;
        for (int k=0; k<26; k++) {
            flag = (key1 * k) % 26;
            if (flag == 1) {
                key1 = k;
                cout << "key 1 : "<< key1 << endl;
                break;
            }
        }
    }

    str_size = strlen(str);
    AffineCeasarCipher(mode, str, str_size, key1, key2);

    if (mode == 1) {
        cout << "[암호화된 결과 출력]" << endl;
        cout << str << endl;
    } else {
        cout << "[복호화된 결과 출력]" << endl;
        cout << str << endl;
    }

    return 0;
}

char *AffineCeasarCipher(int mode, char *str, int str_size, int k1, int k2) {
    // 복호화 과정에서 k1의 inverse값 구하는 과정 삽입하기
    for (int i = 0; i < str_size; i++) {
        // 대문자일 경우
        if ((str[i] >= 'A') && (str[i] <= 'Z')) {
            // 암호화
            if (mode == 1) {
                str[i] = ((k1 * (str[i] - 'A') + k2) % 26) + 'A';
            }
            // 복호화
            else if (mode == 2) {
                int temp = str[i] - 'A' - k2;
                while (temp < 0)
                    temp += 26;
                str[i] = ((k1 * temp) % 26) + 'A';
            }
        }
        // 소문자일 경우
        else if ((str[i] >= 'a') && (str[i] <= 'z')) {
            // 암호화
            if (mode == 1) {
                str[i] = ((k1 * (str[i] - 'a') + k2) % 26) + 'a';
            }
            // 복호화
            else if (mode == 2) {
                int temp = str[i] - 'a' - k2;
                while (temp < 0)
                    temp += 26;
                str[i] = ((k1 * temp) % 26) + 'a';
            }
```

```
        }
        else
            ;
    }

    return str;
}
```



[문제 1 코드 동작 과정]

1. **main()** : mode(암호화 or 복호화), str(암/복호화 할 문자열), k1, k2(두 키) 입력 받기
    a. 복호화 mode일 시, k1의 inverse 계산
    b. AffineCeasarCipher() 함수 호출
2. **AffineCeasarCipher()** : str을 대문자, 소문자로 나누어 암호화 또는 복호화 처리
    a. 문자가 알파벳이 아닌 경우는 무시
    b. 처리된 문자열을 반환
3. **main()** : 처리된 문자열 출력

[문제 2] Gronsfeld cipher는 Shift 키를 숫자로 정의하는 Vigenere 암호의 변형이다. Gronsfeld 암호를 사용하여 평문을 암호문으로 변환하려면 일반 텍스트의 문자를 하나씩 가져 와서 해당 키의 번호에 해당하는 Shift 값을 적용한다. 예를 들어, 암호화 할 텍스트가 "gronsfeld"이고 키가 1234 인 경우 알파벳의 1 자리 G를 이동하여 H가되고 R이 2 자리 이동하고 T가 된다.

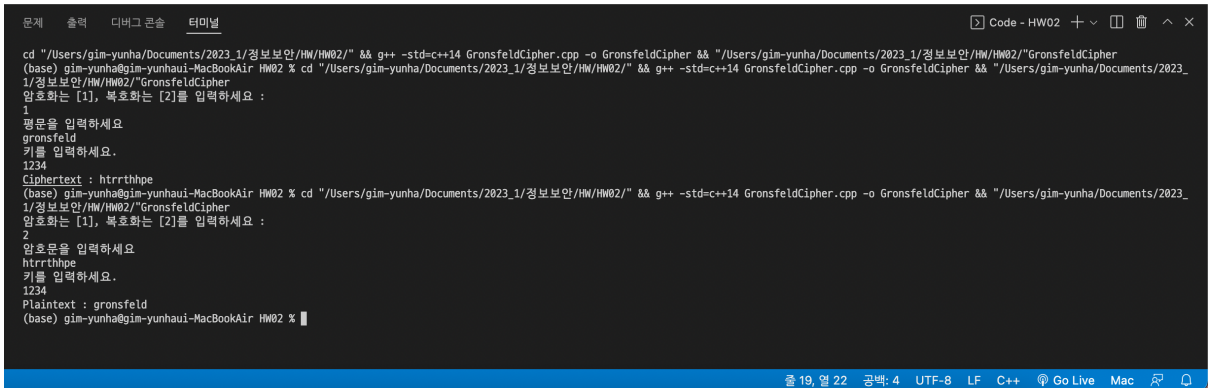
O 예제: Plain text : gronsfeld, Key : 123412341, Cipher text : htrrthhpe

Gronsfeld cipher 소스 코드를 작성하고 동작 과정을 기술하시오.

[문제 2 소스 코드]

```
#include <iostream>
#include <string>

using namespace std;

void Encryption(string, string);
void Decryption(string, string);

int main() {
    int mode;
    string ptext, ctext, key;
    cout << "암호화는 [1], 복호화는 [2]를 입력하세요 :" << endl;
    cin >> mode;
    switch (mode)
    {
    case 1:
    {
        cout << "평문을 입력하세요" << endl;
        cin >> ptext;
        cout << "키를 입력하세요." << endl;
        cin >> key;
        Encryption(ptext, key);
        break;
    }
    case 2:
    {
        cout << "암호문을 입력하세요" << endl;
        cin >> ctext;
        cout << "키를 입력하세요." << endl;
        cin >> key;
        Decryption(ctext, key);
        break;
    }
    default:
    {
        cout << "[INPUT ERROR] 제대로 입력하지 않았습니다." << endl;
        return 0;
    }
    }

    return 0;
}

void Encryption(string plaintext, string key) {
    string ciphertext;
    for (int i=0; i<plaintext.length(); i++) {
        int key_value = key[i % key.length()] - '0';
        int plain_value = (int)plaintext[i];
        int cipher_value = ((plain_value - 32 + key_value) % 94) + 32;
        char cipher_char = (char)cipher_value;
```

```
        ciphertext += cipher_char;
    }
    cout << "Ciphertext : "<< ciphertext << endl;
}

void Decryption(string ciphertext, string key) {
    string plaintext;
    for (int i=0; i<ciphertext.length(); i++) {
        int key_value = key[i % key.length()] - '0';
        int cipher_value = (int)ciphertext[i];
        int plain_value = ((cipher_value - 32 - key_value + 94) % 94) + 32;
        char plain_char = (char)plain_value;
        plaintext += plain_char;
    }
    cout << "Plaintext : " << plaintext << endl;
}
```



[문제 2 코드 동작 과정]

1. **main()** : mode(암호화/복호화) 입력 받은 후, mode에 따라 ptext, ctext, key를 입력받아 Encryption() 또는 Decryption() 함수 호출
2. **Encryption()** : plaintext, key값을 이용해 암호화 작업 수행
    a. 평문 문자열의 길이만큼 반복문 수행.
    b. 평문 문자열의 각 문자와, 해당 순서에서의 키 값을 이용해 암호문 문자열의 각 문자을 생성
        - 암호화 방법 : plain_value와 key_value를 더하고, 이를 94(출력 가능한 ASCII 코드 문자 개수)로 나눈 나머지를 구함. 그 나머지에 32(ASCII 코드 공백 문자) 더해 암호문 생성.
    c. 생성된 암호문 문자열을 출력

3. **Decryption()** : ciphertext, key값을 이용해 복호화 작업 수행
    a. 암호문 문자열의 길이만큼 반복문 수행
    b. 암호문 문자열의 각 문자, 해당 순서에서의 키 값을 이용해 평문 문자열의 각 문자를 생성
        - 복호화 방법 : cipher_value에서 key_value를 빼고, 이를 94(출력 가능한 ASCII 코드 문자 개수)로 나눈 나머지를 구함. 이에 32(ASCII 코드 공백 문자) 더해 평문 생성.
    c. 생성된 평문 문자열을 출력

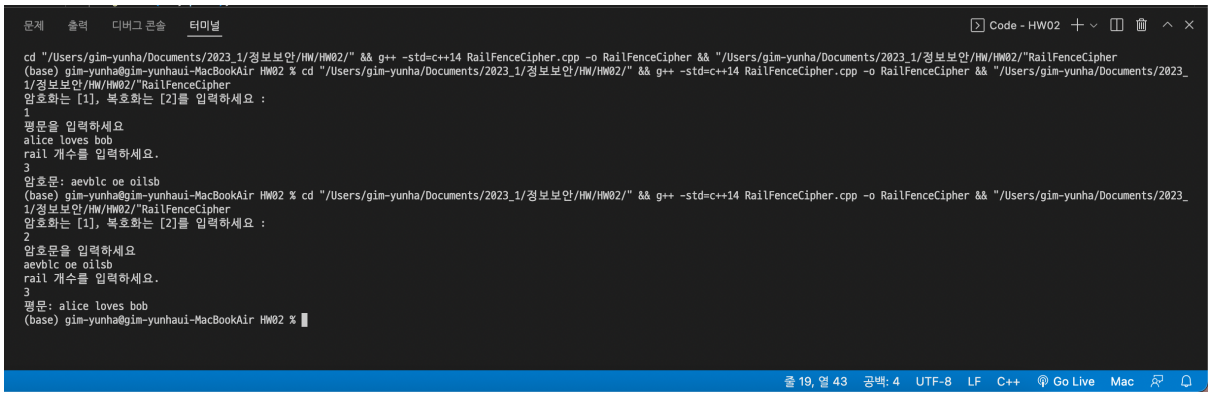
[문제 3] 강의 자료의 Rail Fence cipther 소스 코드를 수정하여 rail의 수가 2이상인 경우에도 동작하는 소스 코드를 작성하고 동작 과정을 기술하시오.

[문제 3 소스 코드]

```
#include <iostream>
#include <vector>
using namespace std;

void Encryption(vector<int>*, string, int);
void Decryption(vector<int>*, string, int);

int main() {

    int mode, rail;
    string ptext, ctext;
    cout << "암호화는 [1], 복호화는 [2]를 입력하세요 :" << endl;
    cin >> mode;

    if (mode == 1) {
        cout << "평문을 입력하세요" << endl;
            cin.ignore();
        getline(cin, ptext);
        cout << "rail 개수를 입력하세요." << endl;
        cin >> rail;
        vector<int>* v = new vector<int>[rail];
        Encryption(v, ptext, rail);
        delete[] v;
    } else if (mode == 2) {
        cout << "암호문을 입력하세요" << endl;
        cin.ignore();
        getline(cin, ctext);
```

```
        cout << "rail 개수를 입력하세요." << endl;
        cin >> rail;
        vector<int>* v = new vector<int>[rail];
        Decryption(v, ctext, rail);
        delete[] v;
    } else {
        cout << "[INPUT ERROR] 제대로 입력하지 않았습니다." << endl;
    }

    return 0;
}

void Encryption(vector<int> *v, string plaintext, int r) {
    string ciphertext;
    int j = 0;
    bool order = false;

    for (int i = 0; i < plaintext.length(); i++) {
        v[j].push_back(i);
        if (j==0 || j==r - 1) {
            order = !order;
        }
        if (order) {
            j++;
        } else {
            j--;
        }
    }
    for (int i = 0; i < r; i++) {
        for (int j = 0; j < v[i].size(); j++) {
            ciphertext += plaintext[v[i][j]];
        }
    }
    cout << "암호문: " << ciphertext << endl;
}

void Decryption(vector<int> *v, string ciphertext, int r) {
    string plaintext(ciphertext.length(), ' ');
    int j = 0;
    bool order = false;
    for (int i = 0; i < ciphertext.length(); i++) {
        v[j].push_back(i);
        if (j == 0 || j == r - 1) {
            order = !order;
        }
        if (order) {
            j++;
        } else {
            j--;
        }
    }
    int idx = 0;
    for (int i = 0; i < r; i++) {
        for (int j = 0; j < v[i].size(); j++) {
            plaintext[v[i][j]] = ciphertext[idx++];
        }
    }
```

```
    cout << "평문: " << plaintext << endl;
}
```



[문제 3 코드 동작 과정]

1. **main()** : mode(암호화/복호화) 입력 받은 후, mode에 따라 ptext, ctext, rail을 입력받아 Encryption() 또는 Decryption() 함수 호출
    a. 벡터는 동적으로 r의 크기만큼 생성
2. **Encryption()** : plaintext, rail 값 이용해 암호화 작업 수행
    a. 평문을 받아와 rail fence의 각 rail vector 별로 평문의 문자를 순서대로 할당
    b. rail이 0 또는 마지막이면 vector 순서를 번갈아가며 지그재그로 할당 가능
    c. 각 rail별로 할당된 문자 읽어와 암호문 만듦.
3. **Decryption()** : ciphertext, rail 값 이용해 복호화 작업 수행
    a. 암호문 받아와 rail fence의 각 rail vector 별로 암호문의 문자를 순서대로 할당
    b. rail이 0 또는 마지막이면 vector 순서를 번갈아가며 지그재그로 할당 가능
    c. 각 rail별로 할당된 문자 읽어와 평문 만듦.