**Classic Cipher 실습**

**[실습 목표] 수업 시간에 학습한 Classic Cipher 소스** 코드를 활용하여 직접 다양한 Classic Cipher프로그램을 제작해봄으로써 현대 암호 알고리즘 동작 원리를 이해하기 위한 학습을 진행한다.

**[유의사항]** 라이브러리를 활용하여 소스 코드를 작성하지 않는다. 동작 결과 화면은 full screen (사용자 실행환경 정보가 보이는 화면. 예: c://yhchoi/a.cpp)을 캡쳐하시오. 각 알고리즘의 동작 과정을 검증하기 위해 강의 자료의 예제를 사용하시오.

**[제출물]** 소스 코드, 소스 코드 동작 과정, 결과 값을 포함하는 보고서를 제출한다.

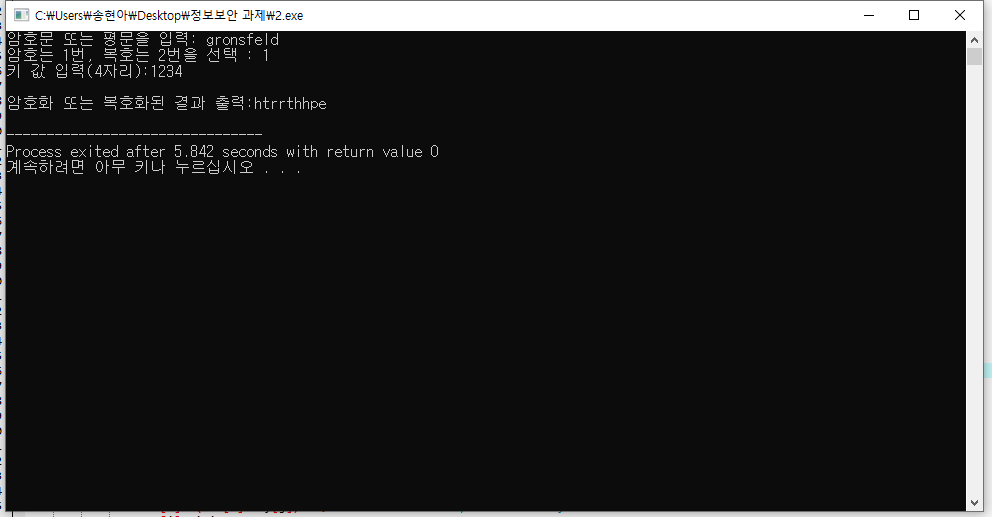
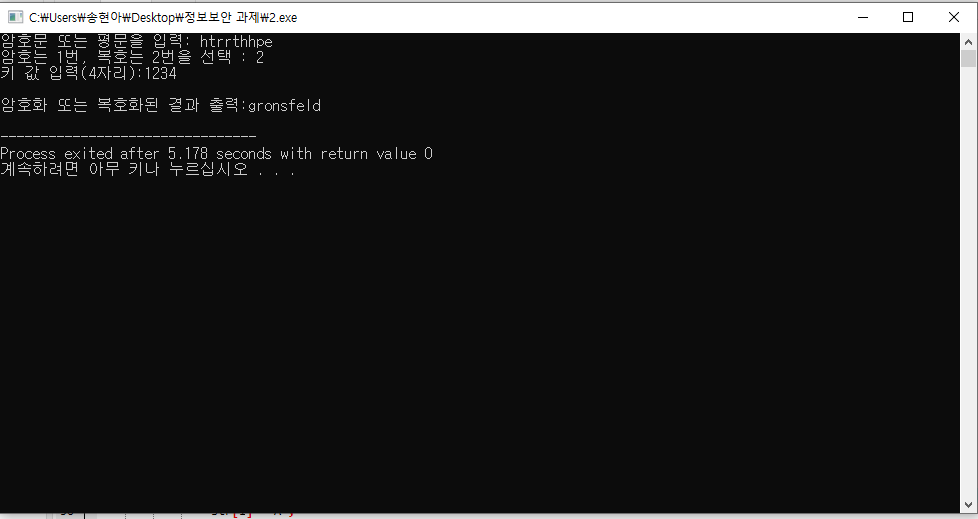
**[문제 1]** 강의 자료 8p의 Affine cipther 소스 코드를 작성하고 동작 과정을 기술하시오.   
Encryption C = (P x K**1** + K**2**) mod 26

Decryption P = { (C - k**2**) x k**1**-1} mod 26

**[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.

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

void Vigenere(int,char\*,int,char\*,int);

int main(){

int k =0, str\_size =0, select=2;

char str[50]={0,};

char key[16]={0,};

printf("암호문 또는 평문을 입력: ");

gets(str); //enter plaintext or ciphertext

printf("암호는 1번, 복호는 2번을 선택 : ");

scanf("%d",&select); //choose encryption or decryption selection

fflush(stdin);

printf("키 값 입력(4자리):");

gets(key); // key stream input

str\_size=strlen(str); // calculate plaintext or ciphertext length

k=strlen(key); //calculate key stream length

Vigenere(select,str,str\_size,key,k); //vigenere cipher function

printf("\n암호화 또는 복호화된 결과 출력:");

puts(str); //encryption or decryption output

return 0;

}

void Vigenere(int select, char\*str, int str\_size, char \*key, int k)

{

int i= 0, j= 0;

for(i=0;i<str\_size;i++){

j=i%k; // key 와 text mapping

if(select ==1){ //암호화 부분

if((str[i]>='a')&&(str[i]<='z')){ // 소문자인 경우

str[i]-='a';

key[j]-='0'; //숫자 아스키 코드 변환

if((str[i]+key[j])<0){ // 음수인 경우

str[i]+=26;

}

str[i]= (str[i]+key[j])%26;

str[i]+='a';

key[j]+='0';

}

else if((str[i]>='A')&&(str[i]<='Z')){ // 대문자인 경우

str[i]-='A';

key[j]-='0'; //숫자 아스키 코드 변환

if((str[i]+key[j])<0){ // 음수인 경우

str[i]+=26;

}

str[i]= (str[i]+key[j])%26; // 암호화의 경우 plaintext + key mod 26

str[i]+='A';

key[j]+='0';

}

}

if (select == 2){ //복호화 부분

if ((str[i] >= 'a') && (str[i] <= 'z')){

str[i] -= 'a';

key[j] -= '0'; //숫자 아스키 코드 변환

if ((str[i] - key[j]) < 0){

str[i] += 26;

}

str[i] = (str[i] - key[j]) % 26; // 복호화의 경우 ciphetext - key mod 26

str[i] += 'a';

key[j] += '0';

}

else if((str[i]>='A')&&(str[i]<='Z')){

str[i]-='A';

key[j]-='0'; //숫자 아스키 코드 변환

if(str[i]-key[j]<0){

str[i]+=26;

}

str[i]= (str[i]-key[j])%26;

str[i]+='A';

key[j]+='0'; //숫자 아스키 코드 변환

}

}

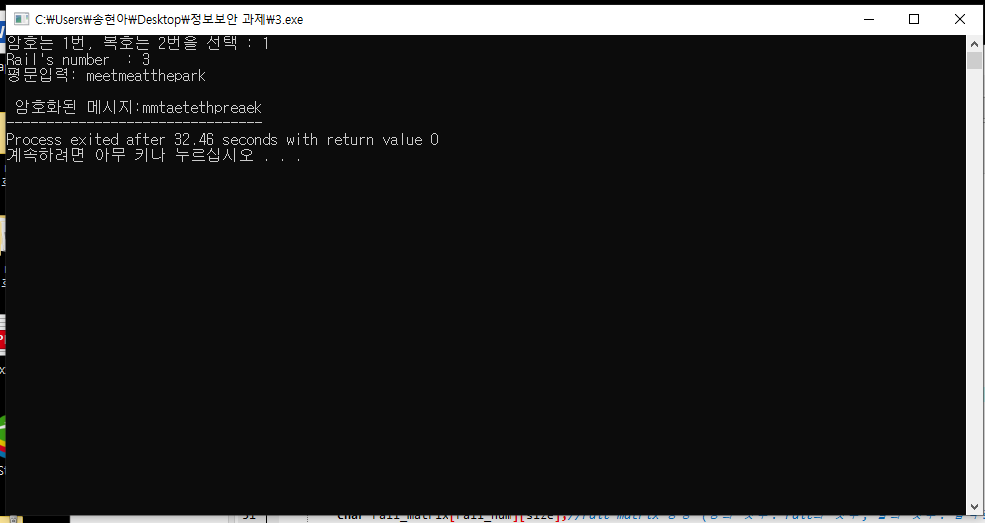
}

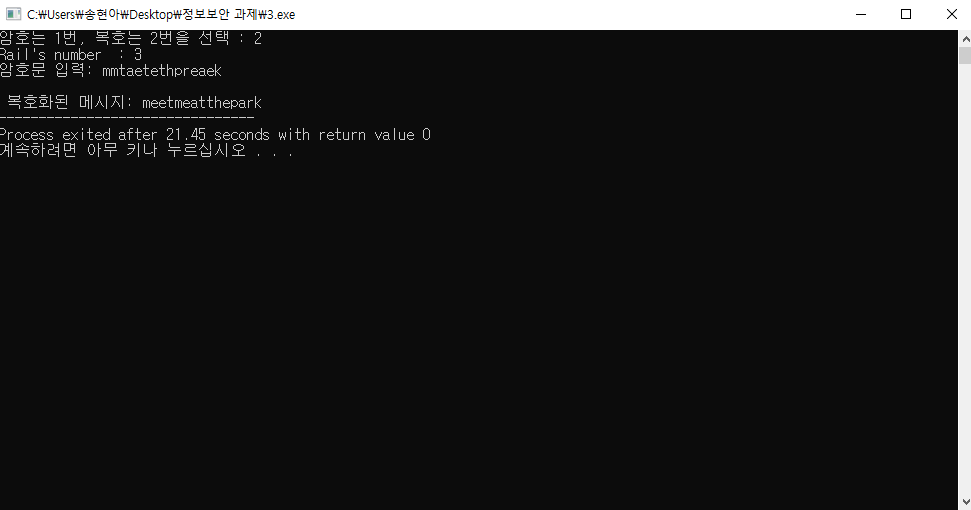
}

[동작과정]

1. 처음에 평문이나 암호문을 문자열로 입력을 받습니다.
2. 암호를 할 것인지 복호화를 할 것인지에 대해 선택을 합니다.
3. 해당 문자열에 적용시킬 키 값을 받습니다. 이 코드에서는 shift를 위해서 숫자를 4자리 입력하도록 하였습니다.
4. 암호화 하는 경우에는 문자열과 키의 index를 매핑해서 같은 자릿수에 있는 값(정수로 바꿈)들끼리 더해서 26으로 나눈 나머지를 더한 후 문자로 반환하여 암호화를 하였습니다.
5. 복호화 하는 경우에는 문자열과 키의 index를 매핑해서 같은 자릿수에 있는 값(정수로 바꿈)들끼리 빼서(ciphetext - key mod 26) 26으로 나눈 나머지를 구해 문자로 반환하여 복호화를 하였습니다.
6. 마지막으로 암호화(또는 복호화)된 결과를 출력합니다.

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





|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **m** | **e** | **e** | **t** | **m** | **e** | **a** | **t** | **t** | **h** | **e** | **p** | **a** | **r** | **k** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**↓**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **m** |  |  |  | **m** |  |  |  | **t** |  |  |  | **a** |  |  |
|  | **e** |  | **t** |  | **e** |  | **t** |  | **h** |  | **p** |  | **r** |  |
|  |  | **e** |  |  |  | **a** |  |  |  | **e** |  |  |  | **k** |