정보보호학개론

기말 실무 과제 발표

기본 암호화 방식 -변형 시저 암호, 비제네르 암호

91714090 송우영
91714284 임주엽

목차

1. 기본 암호화 방식 소개
2. 변형 시저 암호
   - 암호화 코드 소개, 예시
   - 복호화 코드 소개, 예시
   - 변형 시저 암호의 특징과 문제점
3. 비제네르 암호
   - 암호화 코드 소개, 예시
   - 복호화 코드 소개, 예시
   - 비제네르 암호의 특징

기본 암호화 방식

(고전기법)

전치법

단순히 문자열 안의 문자의 위치를 바꾸는 방법

대체법

문자열 안의 문자를 다른 문자로 대체하는 방법

단일 치환 암호

대체하는 문자와 대체되는 문자가 무조건 일대일 대응을 이룬다.
(예:시저 암호)

다중 치환 암호

대체하는 문자와 대체되는 문자가 무조건 일대일 대응을 이루지 않는다.
(예:비제네르 암호)

변형 시저 암호

변형 시저 암호

문자열의 홀수번째 자리와 짝수번째 자리를 따로 암호화 시킴

Ex) 문자열: aaaaaaaa 첫번째 키: 1 두번째 키: 2

→암호문: bcbcbcbc

변형 시저 암호

필요한 변수 선언

암호화 시킬 문장 입력

암호화 키로 쓸 숫자 2개 입력

```c
#include <stdio.h>
#include <string.h>
main()
{
    int i, key1, key2, str_size = 0; //i : 반복문 루프용, key1,key2 : 키값, str_size : 문자열 길이
    char sentence[100]; //암호화 할 문장
    char veri1, veri2;  //키 중복 검사용

    printf("암호화시킬 문장을 입력하세요.₩n");
    gets(sentence);
    str_size = strlen(sentence); //문자열 길이 측정

    printf("첫번째 키로 쓸 숫자를 입력하세요.(음수 가능)₩n");
    scanf("%d", &(int)key1);
    rewind(stdin);   //무한루프 방지를 위한 입력버퍼 제거

    printf("두번째 키로 쓸 숫자를 입력하세요.(음수 가능)₩n");
    scanf("%d", &(int)key2);
    rewind(stdin);

    key1 %= 26;   //키값을 26으로 나눈 나머지로 만듦.
    key2 %= 26;   //key1과 마찬가지

```

변형 시저 암호

키 값 변환시 중복되는 값을 피하기 위해 중복 검사를 함

ex)13과 -13, 4와 -22

```
do
{
    veri1 = 0, veri2 = 0;
    if (veri1 + key1 < 0)
        veri1 += 26;
    if (veri2 + key2 < 0)
        veri2 += 26;
    veri1 = (veri1 + key1) % 26;
    veri2 = (veri2 + key2) % 26;

    if (veri1 == veri2)
    {
        printf("첫번째 키와 같은 값을 가지는 숫자입니다.₩n다른 숫자를 입력하세요.₩n");
        scanf("%d", &key2);
        rewind(stdin);
    }
} while (veri1 == veri2);  //키 중복 검사
```

변형 시저 암호

```
for (i = 0; i < str_size; i += 2)    //홀수번째 암호화
{
    if (sentence[i] >= 'A' && sentence[i] <= 'Z')  //대문자일때
    {
        sentence[i] -= 'A'; //값을 0~25로 만들기 위해 A(65)만큼 뺌.

        if (sentence[i] + key1 < 0) //키 값을 더했을 때 만약 음수라면 26을 더해준다. 암호화 시킬 키값이 음수일 경우에 계산을 위해 넣음.
            sentence[i] += 26;

        sentence[i] = (sentence[i] + key1) % 26;  //키값을 더함.
        sentence[i] += 'A'; //다시 A(65)를 더해줌.
    }

    else if (sentence[i] >= 'a' && sentence[i] <= 'z')  //소문자일때
    {
        sentence[i] -= 'a';

        if (sentence[i] + key1 < 0)
            sentence[i] += 26;

        sentence[i] = (sentence[i] + key1) % 26;
        sentence[i] += 'a';
    }
    else if (sentence[i] == ' ')   //자릿수확인을 위해 띄어쓰기를 _로 대체함.
        sentence[i] = 95;
}
```

홀수번째 자리를 첫번째 키를 이용해 암호화

변형 시저 암호

짝수번째 자리를
두번째 키를 이용해 암호화

```
for (i = 1; i<str_size; i += 2)      //짝수번째 암호화(내용은 홀수번째랑 같음)
{
    if (sentence[i] >= 'A' && sentence[i] <= 'Z')
    {
        sentence[i] -= 'A';

        if (sentence[i] + key2 < 0)
            sentence[i] += 26;

        sentence[i] = (sentence[i] + key2) % 26;
        sentence[i] += 'A';
    }

    else if (sentence[i] >= 'a' && sentence[i] <= 'z')
    {
        sentence[i] -= 'a';

        if (sentence[i] + key2 < 0)
            sentence[i] += 26;

        sentence[i] = (sentence[i] + key2) % 26;
        sentence[i] += 'a';
    }
    else if (sentence[i] == ' ')
        sentence[i] = 95;
}
```

변형 시저 암호

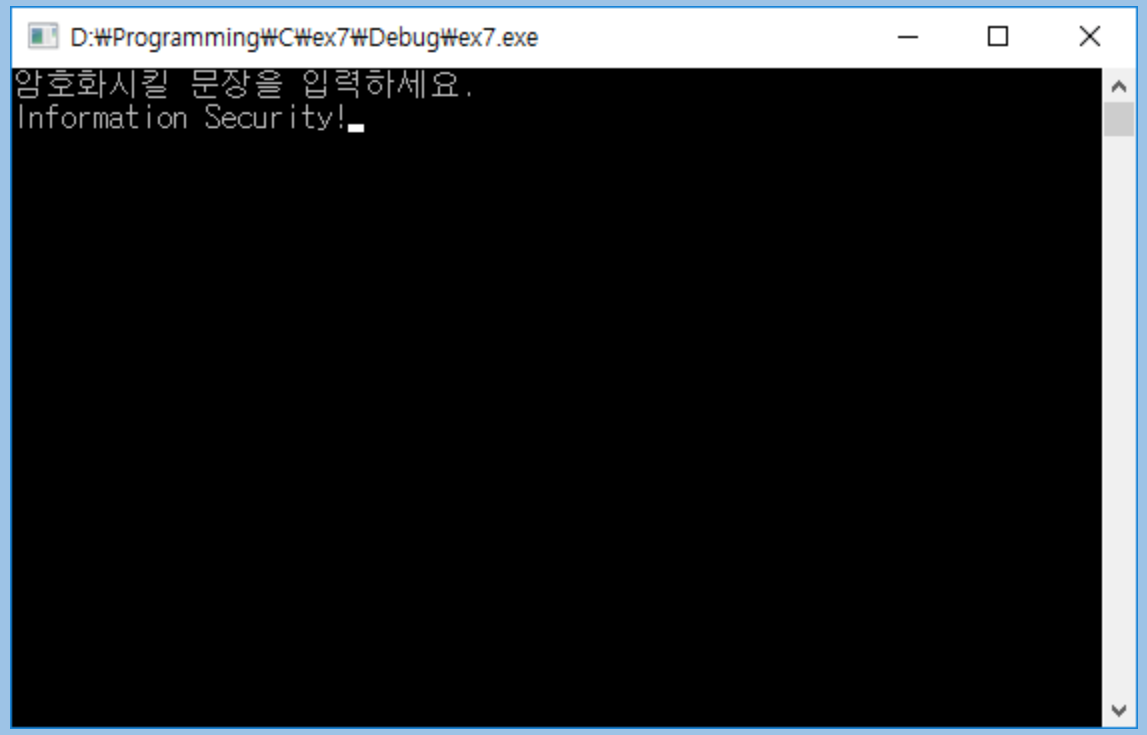
암호화된 문장을 출력함

```
    printf("암호화된 문장:₩n");
    printf("%s₩n", sentence); //암호화 된 값 출력
    getch();   //exe로 실행할 때 바로 꺼지는 것을 방지함.
}
```

변형 시저 암호

실제 암호화

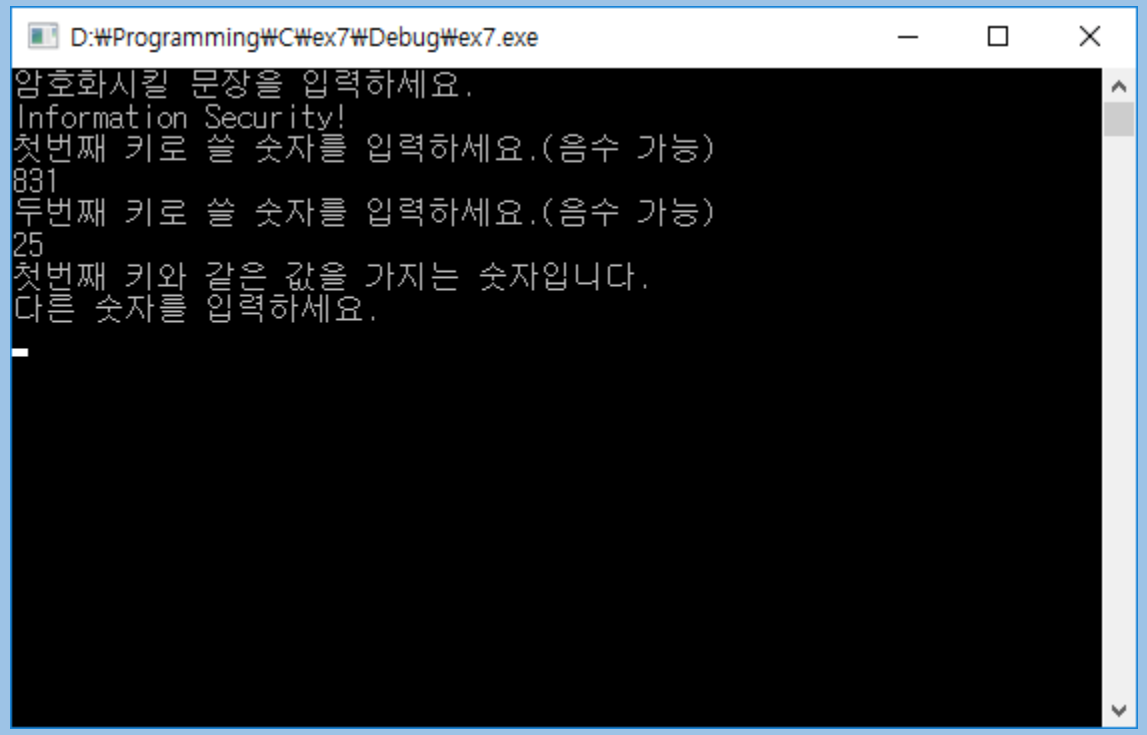
Information Security! 를 입력해봄



변형 시저 암호

실제 암호화

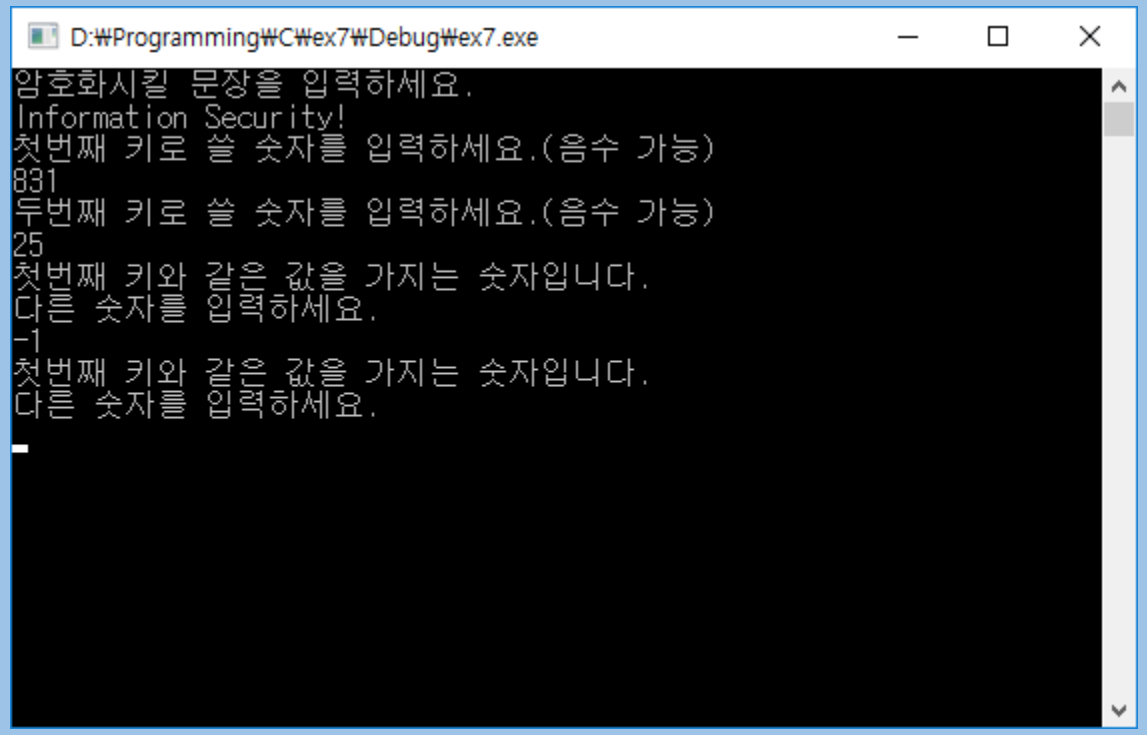
첫번째 키로 831을 넣고
두번째 키로 25를 넣자
중복 값 검사에서 걸림
(831 % 26 = 25)



변형 시저 암호

실제 암호화

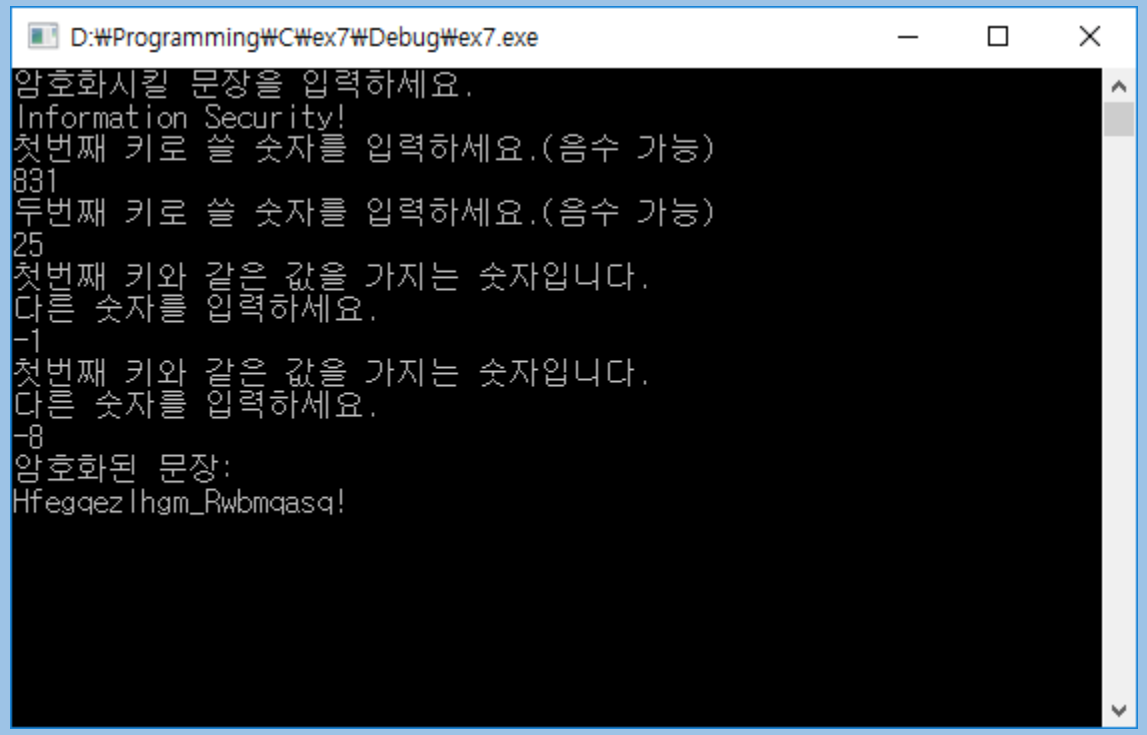
이번엔 25대신에 -1을
넣어주자 다시 한번 중복 값
검사에서 걸림



변형 시저 암호

실제 암호화

첫번째 키로 831을 넣고
두번째 키로 -8을 넣어서
Hfegqezlhgm_Rwbmqasq!
라는 암호화된 문장을 얻음



변형 시저 암호

복호화코드는 암호화 코드에서
문장 암호화를 시키는 코드 부분 위에 이 두 줄만 추가하면 됨

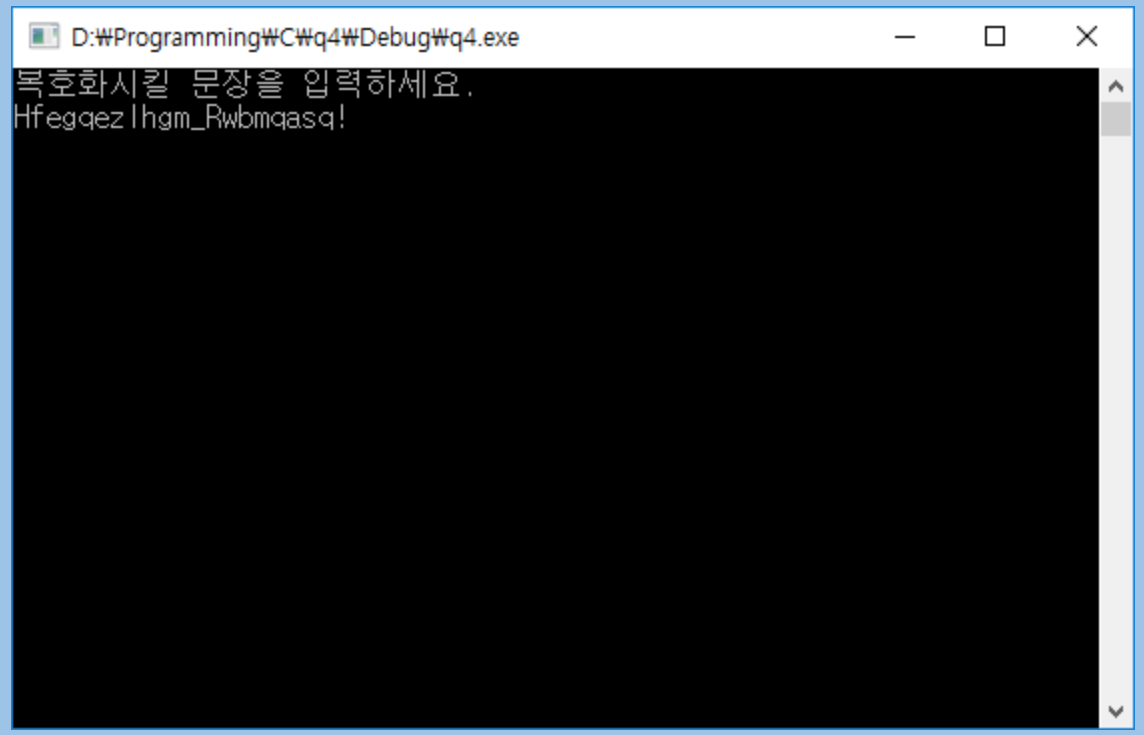
```
key1 = key1*-1;    //복호화를 위해 -1를 곱해서 음수로 만듦.
key2 = key2*-1;    //key1과 마찬가지
```

키 값에 -1을 곱하는 이유: 시저 암호는 복호화를 할 때 암호화 반대로만 하면 되기 때문

변형 시저 암호

실제 복호화

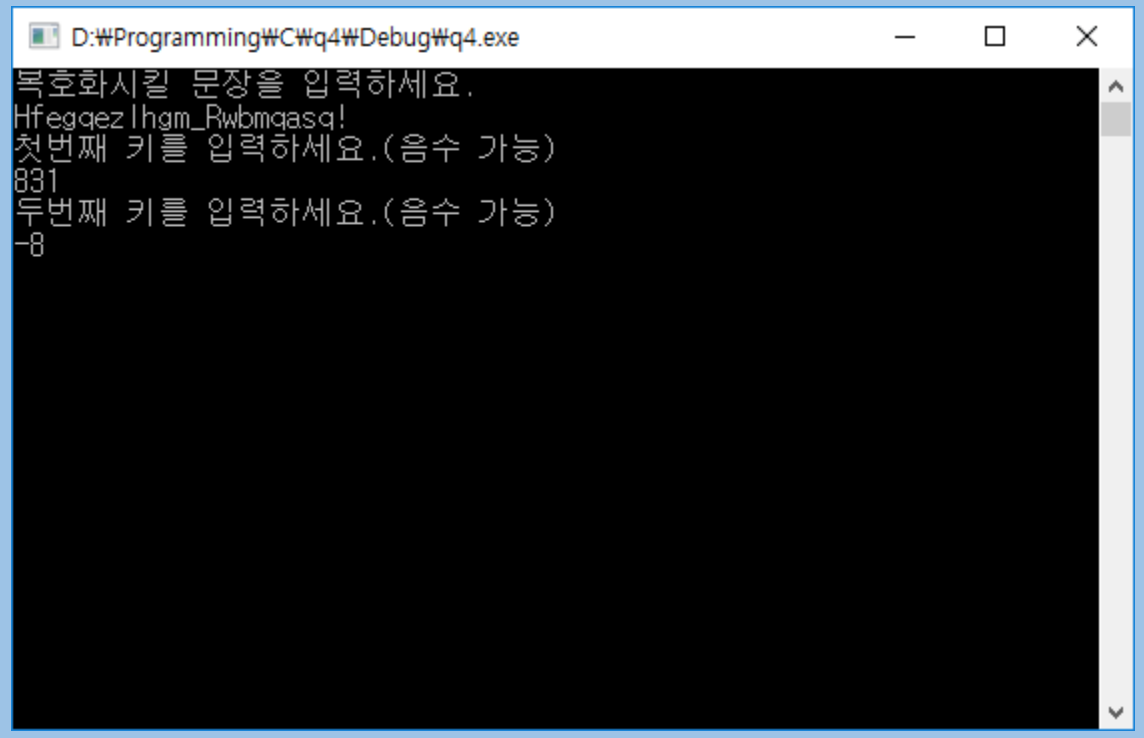
아까 암호화 할때 얻은
Hfegqezlhgm_Rwbmqasq!
라는 문장을 입력함



변형 시저 암호

실제 복호화

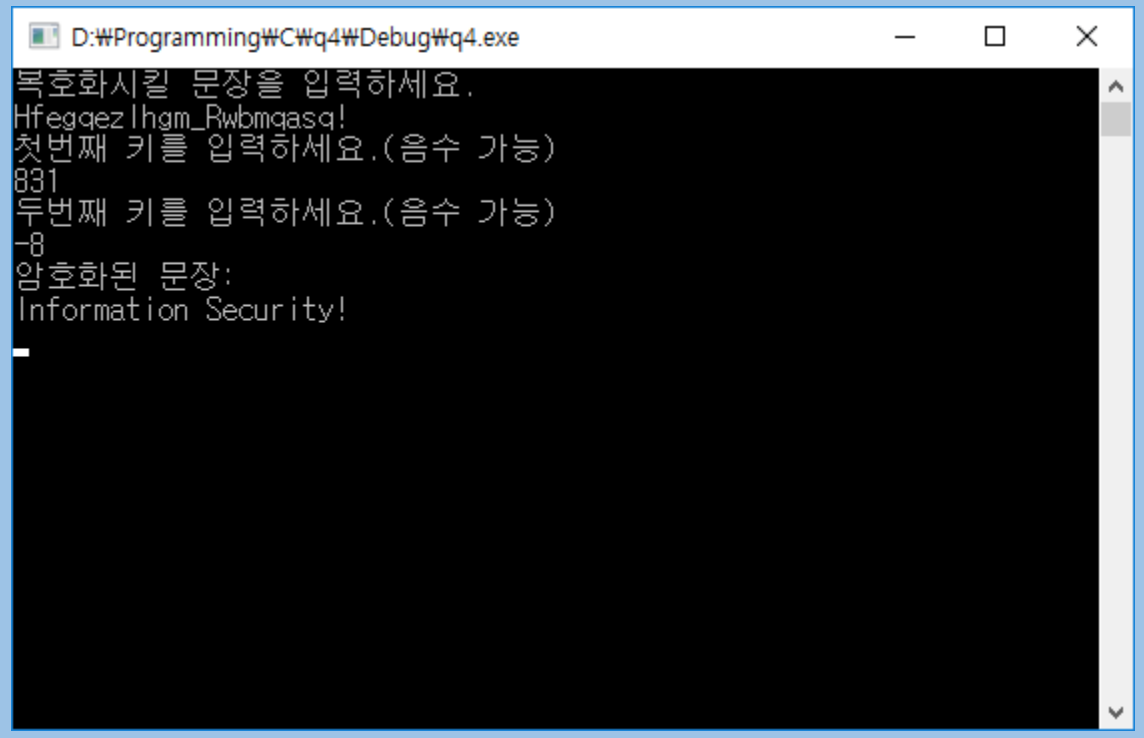
첫번째 키로 쓴 831과
두번째 키로 쓴 -8을 입력함



변형 시저 암호

실제 복호화

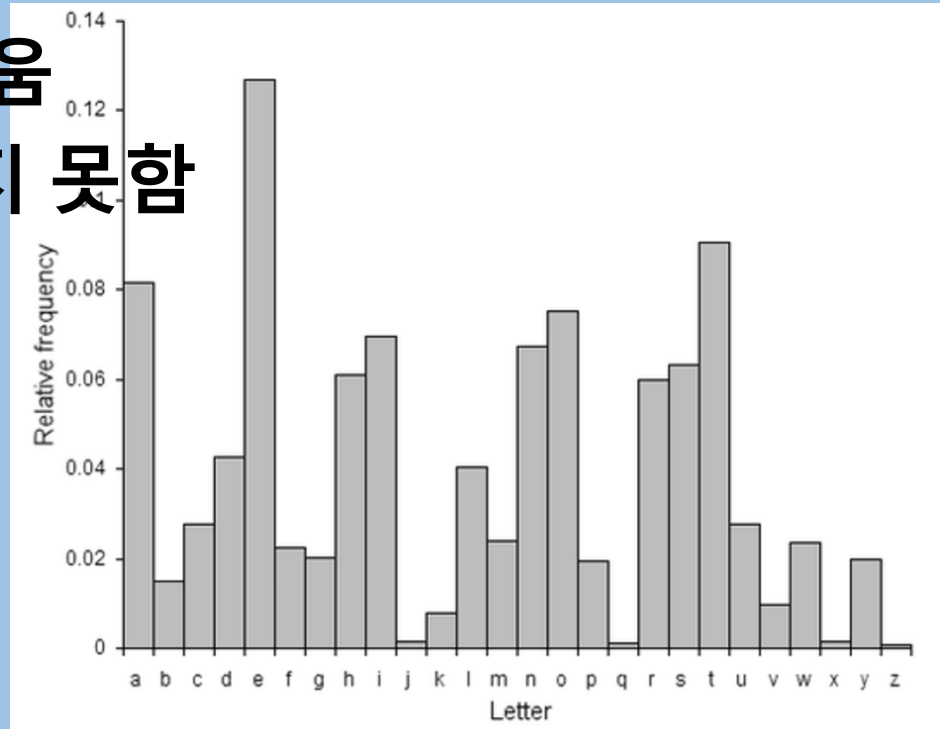
Information Security!가 정상적으로 출력되는 것을 확인 할 수 있음



변형 시저 암호

변형 시저 암호의 특징

- 일반 시저 암호처럼 한자리씩 밀어보는 방법으로는 풀 수 없음
- 마찬가지로 *빈도분석법으로 풀기도 어려움
- 두 키 값, 키의 순서를 모르면 복호화를 하지 못함



***빈도 분석법이란?**
통계적인 방법으로써 알파벳의 출현빈도를 통해 평문의 원래 알파벳을 유추하는 분석법.
단일치환 암호를 해독하기 쉽다.

변형 시저 암호

변형 시저 암호의 문제점

- 암호화할 때 걸었던 키 값이 아니어도 복호화 가능

  key + 26 * x (x는 임의의 정수)일 때 복호화가 가능
- 변형을 해도 일단 베이스가 시저 암호라서 암호화의 강도는 낮음

비제네르 암호

비제네르 암호의 원리

암호 키 문자열: **H e l l o**

암호화 할 문장: **N I c e T o m e e t y o u**

암호화 할 문장의 길이만큼 암호키 문자열을 돌려서 대응

H e l l o H e l l o H e l l o H

N i c e T o M e e t Y o u

U m n p A s X s l x J c b

비제네르 암호

필요한 변수들을 선언하고 암호화 문자키를 입력받은뒤 숫자로 변환해 int 자료형의 keyd 변수에 저장한다.

배열 Keyd에 저장된 알파벳들은 A와a 둘다 0에 대응시킨 순차적인 수열이된다.

```c
#include <stdio.h>
#include <string.h>
main()
{
    char sen[259];
    char key[25];
    int keyd[25];
    int i;
    printf("암호화 문자키를 입력하십시오(24자 이내) : "); gets(key);

    for (int i = 0; i < strlen(key); i++)
    {
        if (key[i] >= 'a'&&key[i] <= 'z')
        {
            keyd[i] = key[i] - 'a';
        }
        else if (key[i] >= 'A'&&key[i] <= 'Z')
        {
            keyd[i] = key[i] - 'A';
        }
        else keyd[i] = key[i] % 26;
    }
```

비제네르 암호

암호화할 문자열을 입력받은뒤 소문자와 대문자일 경우를 나누어 암호화 과정을 실행한다.

```
printf("₩n암호화할 문자열을 입력하십시오. : "); gets(sen);
for (i = 0; i < strlen(sen); i++)
{
    if (sen[i] >= 'a' && sen[i] <= 'z')
    {
        sen[i] = ((sen[i] - 'a' + keyd[i%strlen(key)]) % 26) + 'a';
    }
    if (sen[i] >= 'A' && sen[i] <= 'Z')
    {
        sen[i] = ((sen[i] - 'A' + keyd[i%strlen(key)]) % 26) + 'A';
    }
 }
```

비제네르 암호

```
{
    sen[i] = ((sen[i] - 'a' + keyd[i%strlen(key)]) % 26) + 'a';
}
```

1. 암호화할 문자열의 한문자 sen[i]에서 숫자로 변환된 암호키 keyd와 연산하기 쉽게 a는 0에 대응되도록 a를빼준다.
2. 암호키keyd는 문자열sen의 길이만큼 반복되며 대응되야함으로 Key[i%strlen(key)]으로 계산해 더해준다.
3. 이렇게 계산한 값은 a부터 z까지의 수인 26을 넘을수 있음으로 26을 나눈 나머지를 계산한다.
4. 이값을 다시 알파벳으로 나타내려면 1.에서 빼주었던 a를 다시 더한다.

비제네르 암호

이렇게 짜여진 비제네르 암호 소스코드를 복호하는 소스코드는 단순히 역계산으로 더했던 암호키를 뺌으로써 짜여지면 오류가 난다.

```
sen[i] = (sen[i] - 'a' - keyd[i%strlen(key)]);

if (sen[i] < 0)
{
    sen[i] = (sen[i] + 26) % 26 + 'a';
}
else
{
    sen[i] = sen[i] % 26 + 'a';
}
```

비제네르 암호

일단 암호키를 다시 빼는것까지는 맞지만 문자sen[i]와 암호키keyd[i%strlen(key)]의 차는 음수가 될수 있음으로 if문을 사용해 sen[i]<0일때 알파벳의 개수인 26을 다시 더해줌으로 복호화 코드를 성공적으로 짤수있다.

```
sen[i] = (sen[i] - 'a' - keyd[i%strlen(key)]);

if (sen[i] < 0)
{
    sen[i] = (sen[i] + 26) % 26 + 'a';
}
else
{
    sen[i] = sen[i] % 26 + 'a';
}
```

비제네르 암호

성공적으로
실행이 되었다.

```
암호화 문자키를 입력하십시오(24자 이내) : vigenere

암호화할 문자열을 입력하십시오. : Alpha Bravo Charlie Delta Echo

암호화된 문자
      ↘
      Vtvln Svvdu Plrvgqk Qicxv Kgus

.
.
.
.
.
암호화된 문자를 다시 복호화함.


복호화된 문자
      ↘
      Alpha Bravo Charlie Delta Echo

계속하려면 아무 키나 누르십시오 . . .
```

비제네르 암호

비제네르 암호의 특징

- 빈도분석법으로 해독이 거의 불가능하다.
- 열쇠의 개수가 무궁무진하다.
- 오늘날 전문가들이 많이 사용하는 암호들의 기본형이다.

감사합니다.